12-19-2023

Improved Brain-Storm Optimizer for Disassembly Line Balancing Problems Considering Hazardous Components and Task Switching Time

Ziyan Zhao

Pengkai Xiao

Jiacun Wang

Shixin Liu

Xiwang Guo

See next page for additional authors

Recommended Citation

Zhao, Z.; Xiao, P.; Wang, J.; Liu, S.; Guo, X.; Qin, S.; Tang, Y. Improved Brain-Storm Optimizer for Disassembly Line Balancing Problems Considering Hazardous Components and Task Switching Time. Mathematics 2024, 12, 9. https://doi.org/10.3390/math12010009

mathematics

MDPI

Article

Improved Brain-Storm Optimizer for Disassembly Line Balancing Problems Considering Hazardous Components and Task Switching Time

Ziyan Zhao [1], Pengkai Xiao [2], Jiacun Wang [3], Shixin Liu [1], Xiwang Guo [2], Shujin Qin [4] and Ying Tang [5,*]

[1] College of Information Science and Engineering, Northeastern University, Shenyang 110819, China; zhaoziyan@mail.neu.edu.cn (Z.Z.); sxliu@mail.neu.edu.cn (S.L.)
[2] Information and Control Engineering College, Liaoning Petrochemical University, Fushun 113001, China; xiaopengkai@stu.lnpu.edu.cn (P.X.); x.w.guo@163.com (X.G.)
[3] Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA; jwang@monmouth.edu
[4] College of Economics and Management, Shangqiu Normal University, Shangqiu 476000, China; qinshujin@sqnu.edu.cn
[5] Electrical & Computer Engineering Department, Rowan University, Glassboro, NJ 08028, USA
* Correspondence: tang@rowan.edu

Abstract: Disassembling discarded electrical products plays a crucial role in product recycling, contributing to resource conservation and environmental protection. While disassembly lines are progressively transitioning to automation, manual or human–robot collaborative approaches still involve numerous workers dealing with hazardous disassembly tasks. In such scenarios, achieving a balance between low risk and high revenue becomes pivotal in decision making for disassembly line balancing, determining the optimal assignment of tasks to workstations. This paper tackles a new disassembly line balancing problem under the limitations of quantified penalties for hazardous component disassembly and the switching time between adjacent tasks. The objective function is to maximize the overall profit, which is equal to the disassembly revenue minus the total cost. A mixed-integer linear program is formulated to precisely describe and optimally solve the problem. Recognizing its NP-hard nature, a metaheuristic algorithm, inspired by human idea generation and population evolution processes, is devised to achieve near-optimal solutions. The exceptional performance of the proposed algorithm on practical test cases is demonstrated through a comprehensive comparison involving its solutions, exact solutions obtained using CPLEX to solve the proposed mixed-integer linear program, and those of competitive peer algorithms. It significantly outperforms its competitors and thus implies its great potential to be used in practice. As computing power increases, the effectiveness of the proposed methods is expected to increase further.

Keywords: brain-storm optimization; clustering; disassembly line balancing problem; genetic operators; hazardous disassembly penalty; switching time

MSC: 90-08

Citation: Zhao, Z.; Xiao, P.; Wang, J.; Liu, S.; Guo, X.; Qin, S.; Tang, Y. Improved Brain-Storm Optimizer for Disassembly Line Balancing Problems Considering Hazardous Components and Task Switching Time. *Mathematics* **2024**, *12*, 9. https://doi.org/10.3390/math12010009

Academic Editor: Andrea Scozzari

Received: 17 November 2023
Revised: 16 December 2023
Accepted: 18 December 2023
Published: 19 December 2023

1. Introduction

In the last few years, electronic and electrical products have been updated faster and faster, resulting in a dramatic accumulation of discarded products (also called e-waste). E-waste has reached a staggering amount, but the recycling rate is relatively low, which causes some serious social problems such as resource shortage and environmental pollution. E-waste contains many valuable parts that can be recycled. Recycling is meaningful to not only save resources and materials but also reduce potential environmental pollution. Thus, researching e-waste recycling has become a hot issue in current society. Disassembly is an important step in product recycling. Removing valuable parts from an end-of-life product

through a series of disassembly actions is performed on a disassembly line, which consists of a set of disassembly workstations. How to assign disassembly tasks to workstations of a disassembly line to optimize an/multiple objective function(s) under the condition of satisfying specific constraints is a disassembly line balancing problem (DLBP), which has been widely studied in recent years [1,2].

Some disassembly tasks are risky because the e-waste may have hazardous components, which involve the presence of toxic substances, radioactive materials, or high-voltage components. Since many human workers are employed in manual or human–robot collaborative disassembly lines for dealing with tasks with hazardous parts, the safety of the disassembly process must be considered to avoid accident occurrences that would produce serious social impacts. A low-risk and high-profit disassembly process is the most desired by decision makers [3]. Motivated by this aim, this paper studies a novel linear disassembly line balancing problem, additionally considering the hazardous disassembly penalties and switching time (LDPHS), which has never been studied in previous work. It aims to disassemble parts with high recycling value [4–7] and reduce disassembly risk due to hazardous components. Although some existing studies have dealt with hazardous parts in disassembly line balancing problems [8,9], they usually consider removing them earlier but not their specific penalties. In LDPHS, specific disassembly penalties are given to hazardous parts. It allows decision makers to have a quantitative representation of the hazard degree of different parts. Moreover, the switching time between adjacent tasks is considered in this work, which imposes additional idle time and thus has an important impact on decision making.

DLBPs can be classified into linear [10], parallel [11], bilateral [12], and U-shaped [13] ones according to different disassembly line layouts. Although the last three layouts are the new variants of a linear one, they are rarely used in practice due to two main limitations. One is that they have complicated labor divisions and thus high requirements on the skills of workers and robots. The other is that they bring high pressure on the capital input since each disassembly station requires a complete set of disassembly equipment and tools. Thus, at present, a linear disassembly line is commonly used in practice. It has the advantages of batch disassembly, high disassembly efficiency, clear labor division, and low capital input. Hence, it is necessary and meaningful to study DLBPs on a linear disassembly line to promote their development and progress. The LDPHS considered in this work focuses on a linear layout of the disassembly line.

The methods for solving DLBPs mainly include heuristic algorithms [14], mathematical programming methods [15], and intelligent optimization algorithms [16]. The principles of heuristic algorithms are simple and easy to operate, but the accuracy is limited. Mathematical programming methods can obtain exact solutions for small-scale combinatorial optimization problems [17]. However, due to the NP-hardness of DLBPs [18], large-scale DLBPs cannot be handled effectively by using mathematical programming methods. Intelligent optimization algorithms, such as genetic algorithms [19–22], memetic algorithms [23–26], iterated greedy algorithms [27–30], the teaching-learning-based optimization algorithm [31,32], and some multi-objective ones [33,34], are widely used for solving DLBPs and challenging optimization problems. Brain-storm optimization (BO) is a high-performing intelligent optimization algorithm that was first proposed in [35]. It has been widely applied to solve industrial problems [36,37] in recent research and shows great performance. Its principle mimics the processes of social contact and the idea generation of people, where better ideas are constantly stimulated through the interaction of ideas among people. b Motivated by this, to solve the considered LDPHS, this work designs an improved BO (IBO) by introducing similarity-based clustering and incorporating genetic operators.

The specific contributions of this work are as follows:

(1) It studies a new linear disassembly line balancing problem, which arises from a practical e-waste disassembly process and considers the switching time between tasks and the hazardous disassembly penalty. The new problem is formulated into a mixed-integer linear program that can mathematically describe the concerned problem and exactly solve some limited-scale instances;
(2) It designs an improved brain-storm optimization algorithm to solve large-scale problems where genetic operators and similarity-based clustering are embedded into the framework of a basic brain-storm optimization algorithm. Although the proposed mixed-integer linear program can accurately solve small-scale problems, its performance in solving large-scale problems is very limited. In order to solve large-scale problems in practical industrial applications, it is necessary to design such high-performing approximation algorithms;
(3) It conducts computational experiments to show the high performance of the proposed algorithm. The test cases used in this work are all disassembly cases in practical industrial problems, and their scale ranges from 6 to 46 disassembly tasks. Experimental results show that the proposed algorithm can obtain exact solutions of small-scale cases and high-accuracy solutions of large-scale cases. It significantly outperforms its competitive peers.

The rest of this paper is organized as follows. Section 2 introduces the concerned LDPHS and formulates it into a mixed-integer linear program. Section 3 designs an IBO algorithm. Section 4 reports the experimental results of the proposed model and algorithm to show their effectiveness and great performance in solving the considered problem. Section 5 summarizes this work and discusses future research directions.

2. Problem Description and Modeling

2.1. Problem Description

A disassembly line consists of a set of workstations. A disassembly line balancing problem aims to assign tasks to workstations according to certain criteria. During the disassembly process of e-waste, some disassembly tasks may be hazardous. Hazardous disassembly tasks refer to the process of dismantling components that may pose risks or hazards to the environment, human health, or safety. The hazardous nature of disassembly can arise from the presence of toxic substances, radioactive materials, or high-voltage components. These situations are real in the industrial disassembly of e-waste. We consider the penalties of such hazardous disassembly tasks in the LDPHS, which are called the "hazardous disassembly penalty". The LDPHS aims to assign disassembly tasks to workstations on a linear disassembly line to maximize the total profit. By referring to an actual disassembly process, the total profit is calculated by taking into account the switching cost of adjacent disassembly tasks, the station start-up cost, the hazardous disassembly penalty, and the disassembly cost. Note that the LDPHS deals with an incomplete disassembly process due to the high disassembly risk of some hazardous parts and the high cost of some disassembly operations. There is a fixed cost for turning on a workstation and a sequence-dependent switching time is considered between two adjacent tasks executed on a workstation.

Figure 1 illustrates an example of a discarded electrical product that is to be disassembled on a linear disassembly line, where five tasks are assigned to three workstations that are located linearly. An AND/OR diagram can be used to describe the precedence of disassembly and the conflict relationship in disassembly line balancing problems as shown in Figure 1. It can describe all possible disassembly tasks, subcomponents, and their relationships. It is an effective method to model the priority and conflict relationship among EOL product disassembly tasks. An AND/OR diagram is made up of a series of arrows and super arcs, where arrows indicate the sequence of tasks and super arcs represent the tasks to be executed. Angle brackets in an AND/OR diagram indicate component numbers. Multiple components can be obtained by executing disassembly tasks. By using

an AND/OR diagram, the conflict relation set and priority relation set can be described easily. In an AND/OR diagram, each task is represented by a super arc labeled by integers 1, 2, ..., J, where J is the number of disassembly tasks. The nodes in the figure represent the subcomponents of the product. Each node corresponds to a subcomponent indexed from $\langle 1 \rangle$ to $\langle I \rangle$, where I is the number of all subcomponents. If we obtain a subcomponent q by disassembling subcomponent p, then we define p as the parent subcomponent of q and q as the child subcomponent of p. For example, subcomponent $\langle 2 \rangle$ in the AND/OR graph shown in Figure 1 is disassembled by task 10 to obtain subcomponents $\langle 5 \rangle$ and $\langle 18 \rangle$, and thus, subcomponent $\langle 2 \rangle$ is the parent one, and subcomponents $\langle 5 \rangle$ and $\langle 18 \rangle$ are the subcomponent ones. The super arcs caused by disassembling multiple tasks of the same component are mutually exclusive (i.e., conflict). For example, tasks 10 and 11 conflict with each other. A disassembly incidence matrix $D = [d_{ij}]$ is constructed to define the relationships between subcomponents and tasks:

$$d_{ij} = \begin{cases} 1, \text{If subcomponent } i \text{ can be obtained after task } j; \\ -1, \text{If subcomponent } i \text{ is disassembled via task } j; \\ 0, \text{Otherwise.} \end{cases}$$

A disassembly incidence matrix of the example given in Figure 1 is shown as follows.

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

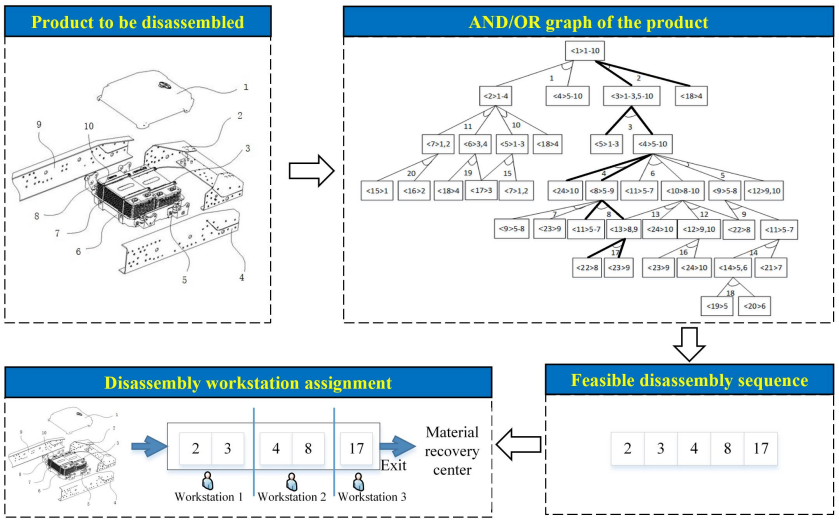

Figure 1. Illustration of a disassembly line balancing problem.

Given an AND/OR graph of a product to be disassembled, a feasible disassembly sequence can be easily obtained. Then, the tasks in the disassembly sequence are assigned to given workstations to maximize the total profit.

2.2. Mathematical Model

To provide a rigorous mathematical description, we formulate LDPHS into a mixed-integer linear program. The related symbols and decision variables used in the model are defined as follows:

Sets:

$\mathbb{J}$: Set of tasks.
$\mathbb{I}$: Set of components.
$\mathbb{W}$: Set of workstations.
$\mathbb{K}$: Set of candidate tasks on each workstation.
$\mathbb{J}_j^C$: Set of tasks conflicting with task j.
$\mathbb{J}_j^P$: Set of the preceding tasks of task j.

Indexes:

j : Disassembly task index, $j = 1, 2, \ldots, J$, where J is the number of tasks.
i : Subcomponent index, $i = 1, 2, \ldots, I$, where I is the number of subcomponents.
k : Candidate position index to arrange tasks, $k = 1, 2, \ldots, K$, where K is the number of candidate tasks on each machine.
w : Workstation index, $w = 1, 2, \ldots, W$, where W is the number of workstations.

Parameters:

T : Work cycle of each workstation.
K : The number of tasks.
W : The number of workstations.
C^S : Task switching cost per unit time.
$T_{j,w}$: Disassembly time of task j at workstation w.
$C_{j,w}$: Disassembly cost of task j at workstation w.
C_w^W : Fixed cost when workstation w is turned on.
$H_{j,w}$: Hazardous penalty for task j on workstation w.
$S_{j,j',w}$: Switching time when tasks j and j' are executed consecutively on workstation w.
P_i : Revenue of component i.

Decision variables:

$$x_{j,w,k} = \begin{cases} 1, \text{ If task } j \text{ is the } k\text{-th task on workstation } w; \\ 0, \text{ Otherwise.} \end{cases}$$

$$y_w = \begin{cases} 1, \text{If workstation } w \text{ is used;} \\ 0, \text{Otherwise.} \end{cases}$$

$z_{w,k}$: The required switching time before performing the k-th task on workstation w.

Based on these symbols and decision variables, a mixed-integer linear program is given as follows:

$$\begin{aligned} \max & \sum_{j\in\mathbb{J}}\sum_{w\in\mathbb{W}}\sum_{i\in\mathbb{I}}\sum_{k\in\mathbb{K}} P_i d_{ij} x_{j,w,k} - \sum_{j\in\mathbb{J}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} C_j x_{j,w,k} \\ & - \sum_{w\in\mathbb{W}}\sum_{k\in K} C^S z_{w,k} - \sum_{w\in\mathbb{W}} C_w^W y_w - \sum_{j\in\mathbb{J}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} H_{j,w} x_{j,w,k} \end{aligned} \tag{1}$$

$$\text{subject to:} \quad \sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} x_{j,w,k} \le 1, \quad \forall j \in \mathbb{J} \tag{2}$$

$$\sum_{j\in\mathbb{J}} x_{j,w,k} \le 1, \quad \forall w \in \mathbb{W}, k \in \mathbb{K} \tag{3}$$

$$\sum_{j\in\mathbb{J}} x_{j,w,k} \ge \sum_{j\in\mathbb{J}}\sum_{k'=1}^{K-1} x_{j,w,k+1}, \quad \forall w \in \mathbb{W} \tag{4}$$

$$\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} x_{j,w,k} + \sum_{j'\in\mathbb{J}_i^C}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{J}} x_{j',w,k} \le 1, \quad \forall j \in \mathbb{J} \tag{5}$$

$$x_{j,w,k} \le \sum_{j'\in\mathbb{J}_j^P}\sum_{w'=1}^{W-1}\sum_{k'\in\mathbb{K}} x_{j',w',k'} + \sum_{j'\in\mathbb{J}_j^P}\sum_{k'=1}^{K-1} x_{j',w,k'}, \quad \forall w \in \mathbb{W}, \forall j \in \mathbb{J} \tag{6}$$

$$z_{w,k} \ge S_{j,j',w}\left(x_{j,w,k-1} + x_{j',w,k} - 1\right), \forall w \in \mathbb{W}, \forall j, j' \in \mathbb{J}, \quad \forall k \in \mathbb{K} \tag{7}$$

$$\sum_{j\in\mathbb{J}}\sum_{k\in\mathbb{K}} x_{j,w,k} \le My_w, \quad \forall w \in \mathbb{W} \tag{8}$$

$$\sum_{j\in\mathbb{J}}\sum_{k\in\mathbb{K}} T_j x_{j,w,k} + \sum_{k\in\mathbb{K}} Z_{w,k} \le T, \quad \forall w \in \mathbb{W} \tag{9}$$

$$x_{j,w,k} \in \{0,1\}, \quad \forall j \in \mathbb{J}, \forall w \in \mathbb{W}, \forall k \in \mathbb{K} \tag{10}$$

$$y_w \in \{0,1\}, \quad \forall w \in \mathbb{W} \tag{11}$$

Objective Function (1) is to maximize the profit of disassembly, which is equal to the disassembly revenue minus the total cost. The total cost includes the disassembly cost, adjacent task-switching cost, workstation start-up cost, and hazardous disassembly penalty. Constraint (2) ensures that a task can only be executed once at most. Namely, a task can only be executed on at most a position of a workstation. Constraint (3) ensures that, at most, only one task can be executed in a position. Constraint (4) guarantees that when there is a task on a disassembly position, i.e., $\sum_{j\in\mathbb{J}}\sum_{k'=1}^{K-1} x_{j,w,k+1} = 1$, there must be a task on its previous position, i.e., $\sum_{j\in\mathbb{J}} x_{j,w,k} = 1$. Constraints (2)–(4) determine the assignment of disassembly tasks to workstations, which needs to further satisfy the conflict constraints (i.e., Constraint (5)) and priority task constraints (i.e., Constraint (6)). Constraint (5) guarantees that conflict tasks cannot be executed at the same time. In other words, if task j is executed, none of the tasks in $\mathbb{J}_j^C$ can be executed. Constraint (6) ensures that a task can be executed only if its preceding task has been executed earlier than it. Constraint (7) calculates the switching time between two immediately adjacent tasks j and j'. Constraint (8) determines whether a workstation is turned on according to the relationships of decision variables $x_{j,w,k}$ and y_w. If any task is assigned to workstation w, i.e., $\sum_{j\in\mathbb{J}}\sum_{k\in\mathbb{K}} x_{j,w,k} = 1$, the workstation has to be turned on, i.e., $y_w = 1$. Constraint (9) limits the work cycle of each workstation to no longer than T. Constraints (10) and (11) are the ranges of the decision variables.

3. Proposed Algorithm

Recently, many bionics algorithms [38,39] have been designed to solve disassembly line balancing problems. In this paper, an improved brain-storm optimization (IBO) algorithm is designed to solve the concerned LDPHS. A brain-storm optimization algorithm is derived from Smith's simulation of the process of human brainstorming meetings [35], including three iterated stages, i.e., clustering, new individual generation, and selection. Brain-storm optimization algorithms have been widely used in solving industrial optimization problems because of their simplicity and easy implementation [40]. However, to the authors' best knowledge, they have rarely been attempted as a means to solve DLBPs [41]. Motivated by their success in other optimization problems, we adopt and adapt one for solving the LDPHS. In addition to the basic framework of a brain-storm optimization algorithm, the presented IBO integrates genetic operators, i.e., crossover and mutation operators, to assist the process of new idea generation of a basic brain-storm optimization algorithm.

3.1. The Procedure of IBO

The presented IBO has an easy framework as illustrated in Figure 2, which includes four main steps:

(a) Initial population generation: Monte Carlo random simulation is conducted to generate a disassembly task set. The conflict tasks are not allowed to be included in the set for feasibility. Then, the tasks in the set are sorted according to their precedence relationship to form a feasible solution. This step imitates different ideas from people in different fields.

(b) K-means-based clustering: The individuals are clustered according to similarity degrees. Their average similarity degree is set to a cluster center. Specifically, a *K*-means clustering algorithm [42] is used to cluster the individuals. The center is the local optimal solution of each cluster. The best solution obtained in each iteration can be determined by comparing each cluster center. This step mimics a process of gathering people with similar ideas.

(c) New individual generation: Genetic-based individual generation operators, including crossover and mutation, are used to generate new individuals. This step mimics different people coming together to brainstorm new ideas.

(d) Selection and updating: The local optimal solution obtained in the generation is selected and used to update the global optimal solution in all generations.

Repeat steps (b)∼(d) until the maximum number of iterations is reached. During the iteration, the incident matrix introduced in Section 2 is used to judge the feasibility of the solutions. For a feasible solution, objective Function (1) is used to calculate its fitness, i.e., total profit, since the values of all decision variables can be known for a specific individual.

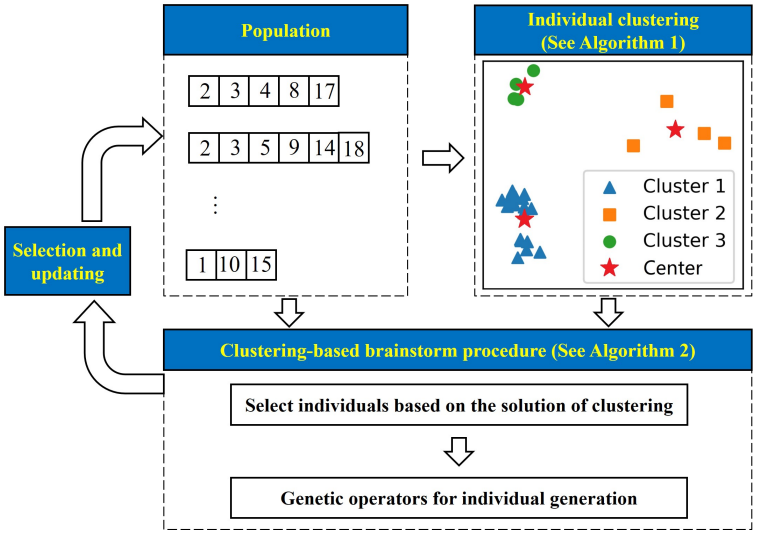

Figure 2. Procedure of IBO.

3.2. Clustering Phase

A *K*-means clustering algorithm [43] is used in IBO, which can cluster individuals with similar characteristics in the short term. In it, the similarity of two individuals is related to the length of their identical segments and the total lengths of their coding. Specifically, for two individuals with lengths m and n, respectively, who share an identical sequence of length I, their similarity can be quantified as $I/max(n,m)$. The distance (denoted as d) between the two individuals is calculated as $d = 1 - I/max(n,m)$. For example, for two individuals, 1→3→5→7 and 1→3→13, $m = 4$, $n = 3$, the same part of the two individuals is 1→3, thus $I = 2$, the similarity of the two individuals is $2/max(4,3) = 0.5$, and $d = 1 - 0.5 = 0.5$. Having the distances between any two of the individuals, K individuals are randomly selected from the population whose fitness values are used as cluster centers, where K is a parameter to be set. Other individuals are divided into different clusters according to the distance from the cluster centers. Then, the average fitness value of all individuals in a cluster is calculated and used as a new cluster center. This clustering procedure continues until cluster centers are not changed anymore. The specific operation is shown in Algorithm 1, in which A and B are two solutions, A_i and A_{i+1} are two sequential tasks in A, and B_i and B_{i+1} are two sequential tasks in B.

Algorithm 1: Clustering

Input: Initial population, cluster count K.
Output: Clusters
Begin
Take out two individuals I^1 and I^2 each time and set $I = 0$.
Compare the elements at each position of I^1 and I^2.
while (I^1 !=NULL and I^2 !=NULL) **do**
if (I_i^1,I_{i+1}^1)==(I_j^2,I_{j+1}^2) **then**
I^1 and I^2 contains the same disassembly task.
Record the occurrence times of this situation by $I = I + 1$.
end if
Calculate the distance d between I^1 and I^2.
end while
Randomly assign K individuals as cluster centers.
for i = 0 to N **do**
// N is the individual count
if The center of a cluster L is closest to i **then**
Arrange i in cluster L.
Calculate all individual fitness in each cluster.
The average of individual fitness is defined as a new cluster center.
end if
end for
End

3.3. Clustering-Based Brainstorm

New individual generation relies on a clustering-based brainstorm procedure, which includes five operators: random thinking, cluster center independent thinking, ordinary individual fusion thinking, cluster center fusion thinking, and ordinary individual independent thinking.

- Random thinking refers to the random generation of an individual. If the individual is better than a cluster center, then replace it.
- Cluster center independent thinking refers to a mutation operation conducted on a randomly selected cluster center, to jump out of a local optimal solution.
- Ordinary individual independent thinking refers to a mutation operation conducted on a randomly selected individual. If the new individual is better than the original one, then replace it.
- Cluster center fusion thinking refers to a crossover operation conducted on two randomly selected cluster centers to achieve fusion.
- General individual fusion thinking refers to a crossover operation conducted on two randomly selected individuals.

The specific execution process of the clustering-based brainstorm procedure is shown in Algorithm 2. In it, p_1 is a probability for determining that a new individual is generated from one or two ordinary individuals, p_2 is a probability for determining that a new individual is generated from one cluster center or an ordinary individual, and p_3 is a probability to determine that a new individual is generated from two cluster centers or two ordinary individuals.

Algorithm 2: Clustering-based brainstorm procedure

Input: Population and clusters, p_1, p_2, p_3.
Output: New individuals.
Begin
for i = 0 to population size **do**
Randomly generate a real number $r_g \in [0,1)$.
if $r_g < p_1$ **then**
Select a cluster randomly along with generating a random value $r_o \in [0,1)$.
if $r_o < p_2$ **then**
Randomly select a cluster center and then generate a new individual via mutation.
else
Randomly select an individual and then generate a new one via mutation.
end if
else
Randomly generate a value $r_t \in [0,1)$.
if $r_t < p_3$ **then**
Randomly select two cluster centers and then get two new individuals by adopting crossover.
else
Randomly select two individuals and then get two new ones by adopting crossover.
end if
end if
Gain the newly formed individuals
end for
End

3.4. Genetic Operators

Crossover and mutation operators in genetic algorithms are adopted to generate new individuals in the aforementioned clustering-based brainstorm. A crossover operator is used to simulate the strategy of generating new individuals based on two previous individuals by crossing their genes. A mutation operator is used to simulate the strategy of generating new individuals by changing one or more genes based on a previous individual.

(a) Crossover operator: To increase the diversity of solutions, a problem-specific crossover operator is designed in IBO. Multiple intersection points of two paternal chromosomes are randomly selected and the intersection points of the two chromosomes are exchanged to generate new chromosomes. Figure 3 shows an example of a crossover process. In it, the selected intersection points are 2, 3, and 6. The tasks of intersection points in the two individuals are 10, 5, 14 and 3, 4, 14, respectively. According to them, each original disassembly sequence is split into two subsequences. Then the obtained subsequences are crossed over and merged to obtain the new individuals.

(b) Mutation operator: In order to further enhance the diversity of the population, a new mutation operator is designed under the premise of ensuring its feasibility. A mutation position is randomly selected on a chromosome. Genes from this position are inserted randomly anywhere on the chromosome. Figure 4 shows an example of a mutation process. In it, the fourth gene in a parent individual is selected and inserted into the next position.

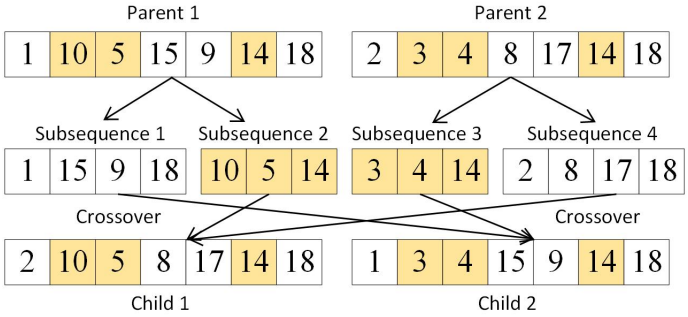

Figure 3. An example of crossover operator.

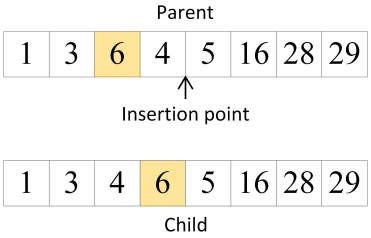

Figure 4. An example of mutation operator.

4. Computational Experiments

4.1. Benchmark Data

Five actual and benchmark disassembly instances with different scales are used for our experiments including a lamp [41], a washing machine [44], a car battery, a radio [45], and a hammer drill [46]. The parameters of the objective function such as the revenue, cost, and penalty are collected and adapted from practical industrial settings. The desensitized data are available on request. The number of disassembly tasks J and the number of components I determine the scale of an instance. The specific scales of the test cases are shown in Table 1. For the largest-scale case, i.e., case 5, it contains 46 disassembly tasks and 62 components.

Table 1. Scales of the test cases.

Case	Product	Number of Disassembly Tasks	Number of Components
1	Lamp	6	7
2	Washing machine	13	15
3	Car battery	20	24
4	Radio	30	29
5	Hammer drill	46	62

4.2. Compared Methods

Numerous experiments were conducted to test the performance of the proposed algorithm. Its solutions were compared with the optimal solutions and those obtained by four newly proposed competitive metaheuristic algorithms including the Grey Wolf Optimizer (GWO) [47], the Whale Optimization Algorithm (WOA) [48], the Dingoes-hunting Optimization Algorithm (DOA) [49], and the Carnivorous Plant Algorithm (CPA) [50]. The optimal solutions were obtained by solving the proposed mathematical model with CPLEX v.12.6.3 under default parameters and configurations. The algorithms were coded in Java and implemented on a computer with the configuration: Intel(R) Core(TM) i7-11800H @ 2.30 and Windows 10 operation system.

When using CPLEX to solve an instance, we limited its running time to 3600 s. The parameter settings of IBO refer to the reference [3]. Specially, the population size was 10,

the number of clusters was 5, p_1 = 0.8, p_2 = 0.4, p_3 = 0.5, and the maximum number of iterations was 500. The compared algorithms used the same population size and iteration times for a fair comparison. Their specific parameter calibrations and settings refer to the existing literature, which can be found in [47,49–51].

4.3. Ablation Study

Genetic operators and a similarity-based clustering mechanism were specifically designed for the problem in the proposed IBO. Before comparing the IBO with its competitors, we first conducted an ablation study to verify the effectiveness of them. In it, we removed the crossover operator, mutation operator, and similarity-based clustering mechanism from IBO, respectively, resulting in three variants of IBO named IBO-C, IBO-M, and IBO-S. We compared the performance of IBO and the three variants by comparing the results obtained after each of the four comparison algorithms independently ran 20 times to solve the largest-scale case, i.e., case 5. The comparison is shown in Figure 5. It can be seen that IBO significantly outperforms its variants both in accuracy and stability. Hence, we conclude that the genetic operators and similarity-based clustering mechanism all play a key role in IBO. It is their combined effect that makes IBO show excellent performance.

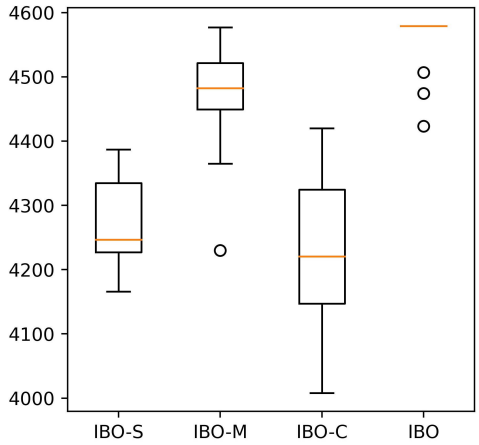

Figure 5. Ablation study.

4.4. Performance Metrics

When comparing IBO with its competitive peers, their accuracy and efficiency are compared by using four performance metrics:

(1) The objective function value, i.e., the total profit of a solution, obtained by a solution method.
(2) The gap to the solution obtained by CPLEX. This is used to evaluate the accuracy of an algorithm. CPLEX can optimally solve the proposed mixed-integer linear program for some cases in our experiments. The exact solutions obtained by it are marked in bold in the experimental results (to be introduced later). Let Obj_{Alg} and Obj_{CPLEX} mean the objective function values of two solutions obtained by an algorithm and CPLEX, respectively. This metric is calculated by using $(Obj_{Alg} - Obj_{CPLEX})/Obj_{CPLEX} * 100\%$.
(3) The gap to the solution obtained by IBO. This is used to validate the superiority of the proposed algorithm to its competitors. It compares the solution obtained by an algorithm with the one obtained by our algorithm. Let Obj_{Alg} and Obj_{IBO} mean the objective function values of two solutions obtained by a competitor and IBO, respectively. This metric is calculated by using $(Obj_{IBO} - Obj_{Com})/Obj_{Com} * 100\%$.
(4) Running time. This is used to evaluate the efficiency of an algorithm. The average running time of our algorithm for solving a case is reported and compared with those compared methods.

4.5. Performance Comparison

The experimental results for the test cases are compared in Tables 2–6. After analyzing the results from them, we present the following discussions:

(1) Solving the proposed mathematical model directly by using CPLEX can obtain the optimal solutions for cases 1–4 in limited time, which verifies the effectiveness of the presented model. However, for a large-scale instance, i.e., case 5 reported in Table 6, CPLEX struggles to find its optimal solution within the limited time since CPLEX's performance can degrade as the scale of an optimization problem increases. As a combinatorial optimization problem, the concerned LDPHS is NP-hard, which requires exponential time to find its optimal solution. Thus, a large-scale case with a large number of decision variables and constraints becomes computationally expensive and cannot be optimally solved within the limited time. Note that, for case 5, the total profit (2564) obtained by CPLEX is the best-found feasible solution (i.e., the lower bound) and the relaxed solution reported by CPLEX (i.e., the upper bound) is 7661. Moreover, CPLEX reports that it takes 49 s to find a solution for case 5 but cannot find the optimal solution within an hour. Typically, a solution time in the order of minutes is acceptable in industrial scenarios [52]. Numerous existing studies also emphasize the necessity of making decisions on analogous disassembly line balancing problems within minutes [18,53,54]. In addition, the cycle time in disassembly line balancing problems generally resides within the realm of minutes [46,55], which underscores the need to make timely decisions. Thus, it is considerably time-consuming for practical industrial application scenarios to solve the mathematical model by CPLEX. In addition, huge fluctuations in the solution time of CPLEX imply it is not too unreliable to be directly used in industry, and thus indicate the need to design metaheuristics.

(2) For an instance with a small size such as cases 1 and 2, the presented IBO can obtain the same solution as that obtained by CPLEX, which means that it can optimally solve them. For medium-scale instances such as cases 3 and 4, although the presented IBO cannot obtain the optimal solutions that CPLEX can, it can obtain a competitive solution with a relatively low gap in a short time that is much faster than CPLEX. For a large-scale instance, i.e., case 5, which cannot be optimally solved by using CPLEX, the presented IBO can solve it very quickly and obtain a great solution (with a 44.01% improvement), showing that it has a great advantage over competitors.

(3) Case 1 is the simplest case, which can be optimally solved by all of the compared algorithms. Thus the gaps reported in Table 2 are 0% for all the algorithms. When solving case 2, IBO, CPA, and WOA can still obtain optimal solutions, while GWO and DOA cannot, with gaps of 2.66% and 5.39% to the optimal solution as reported in Table 3. For cases 3–5, IBO shows the best performance and obtains much better solutions than those of its peers. These indicate its great advantages in accuracy.

(4) The running time of the compared metaheuristics algorithms increases with the scale of the cases. When solving the same case, the time consumption of different algorithms is very close. To clearly compare their time complexity, we draw their time consumption curves in Figure 6. It shows that IBO and its competitors have similar time complexities. When solving the same case, no algorithm has a clear advantage in running time. As the problem size increases, the running time of IBO and its competitors gently increases. The largest case, i.e., case 5, takes only a few seconds to solve, implying that IBO is fast enough to solve large-scale problems.

(5) All the test cases can be solved by using IBO in 7 s. Its high performance and low running-time consumption imply its great potential to be used in practice.

In addition, a statistical comparison [56] of the algorithm was performed to illustrate the advantage of IBO over its competitors. The solutions obtained by 20 independent runs of each algorithm for solving the test cases were used to draw the box plots in Figure 7. Since the exact solutions of small-scale cases can be easily obtained, we only draw the box plots of the results for cases 3–5 in Figure 7. It can clearly be seen that

IBO shows statistical advantages in the accuracy and stability, which further indicates its high performance.

Table 2. Results comparison of disassembling a lamp (Case 1).

	Total Profit	Gap to the Solution Obtained by CPLEX	Gap to the Solution Obtained by IBO	Running Time (s)
CPLEX	**516**	−−	0%	0.14
GWO	**516**	0%	0%	0.17
DOA	**516**	0%	0%	0.23
CPA	**516**	0%	0%	1.03
WOA	**516**	0%	0%	0.33
IBO	**516**	0%	−−	0.19

Note: Bold indicates that the maximum total profit among the compared algorithms.

Table 3. Results comparison of disassembling a washing machine (Case 2).

	Total Profit	Gap to the Solution Obtained by CPLEX	Gap to the Solution Obtained by IBO	Running Time (s)
CPLEX	**1349**	−−	0.00%	1.57
GWO	1314	2.66%	2.66%	0.52
DOA	1280	5.39%	5.39%	0.72
CPA	**1349**	0.00%	0.00%	1.32
WOA	**1349**	0.00%	0.00%	0.65
IBO	**1349**	0.00%	−−	0.50

Note: Bold indicates that the maximum total profit among the compared algorithms.

Table 4. Results comparison of disassembling a car battery (Case 3).

	Total Profit	Gap to the Solution Obtained by CPLEX	Gap to the Solution Obtained by IBO	Running Time (s)
CPLEX	**1849**	−−	−16.44%	277.45
GWO	1351	36.86%	14.92%	0.84
DOA	1356	36.36%	14.61%	1.36
CPA	1376	34.38%	13.35%	2.30
WOA	1361	35.86%	14.29%	1.36
IBO	1588	16.44%	−−	0.99

Note: Bold indicates that the maximum total profit among the compared algorithms.

Table 5. Results comparison of disassembling a radio (Case 4).

	Total Profit	Gap to the Solution Obtained by CPLEX	Gap to the Solution Obtained by IBO	Running Time (s)
CPLEX	**3420**	−−	−2.03%	310.77
GWO	2952	15.85%	11.93%	2.38
DOA	3050	12.13%	9.01%	2.65
CPA	3062	11.69%	8.65%	2.63
WOA	3314	3.20%	1.13%	3.24
IBO	3352	2.03%	−−	2.53

Note: Bold indicates that the maximum total profit among the compared algorithms.

Table 6. Results comparison of disassembling a hammer drill (Case 5).

	Total Profit	Gap to the Solution Obtained by CPLEX	Gap to the Solution Obtained by IBO	Running Time (s)
CPLEX	2564	−−	44.01%	3600.00
GWO	4484	−42.82%	2.07%	7.44
DOA	4332	−40.81%	5.39%	4.55
CPA	4486	−42.84%	2.03%	5.63
WOA	4349	−41.04%	5.02%	6.60
IBO	4579	−44.01%	−−	6.84

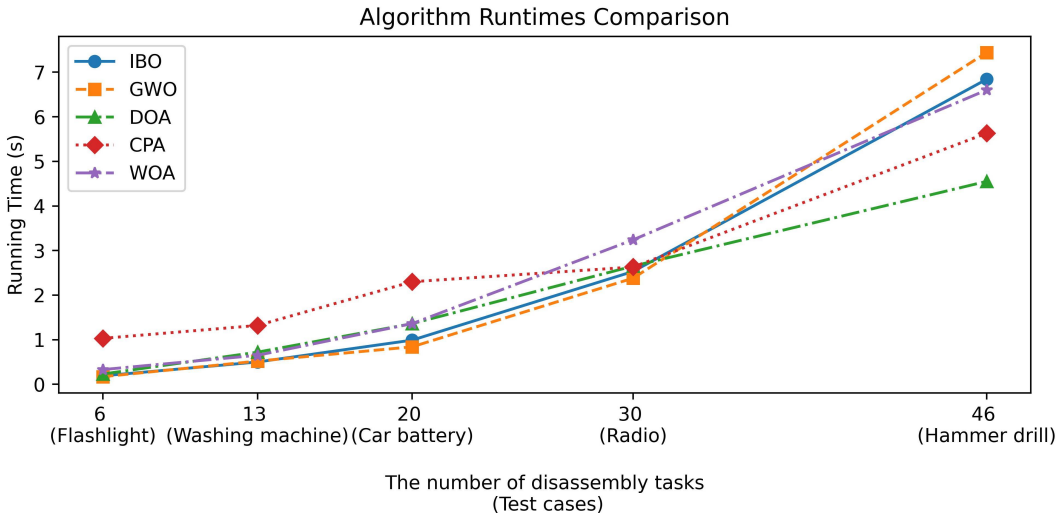

Figure 6. Running time comparison.

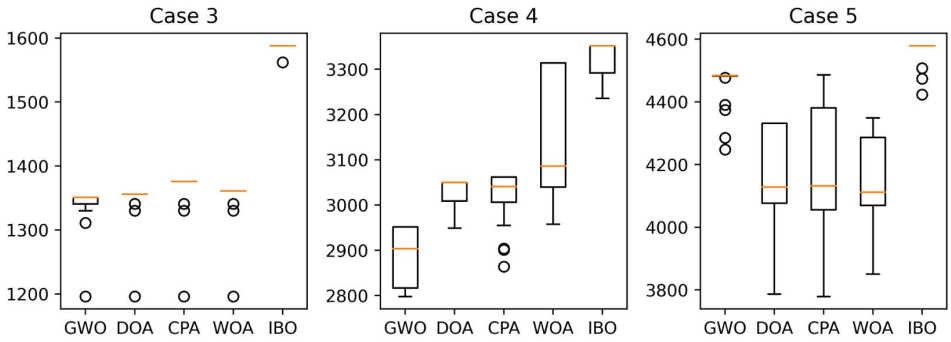

Figure 7. Statistical comparison of the algorithms.

In addition, to clearly show the convergence of the compared metaheuristic algorithms, their convergence curves for solving cases 2 and 5 are given in Figures 8 and 9. They show that IBO shows great performance in the early stage of its iteration and achieves convergence relatively early compared with the other algorithms. These further indicate the superiority of IBO compared with its competitors in solving the concerned problem.

According to the above analyses, we can draw the conclusion that the proposed mixed-integer linear program is an effective mathematical model and IBO outperforms its competitive peers in solving the concerned problem, which has great application potential. These results justify the contributions of this work.

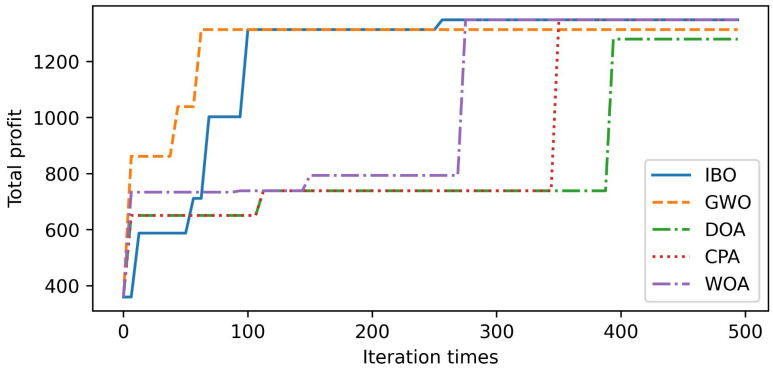

Figure 8. Comparison of IBO algorithm with other solution methods on case 2.

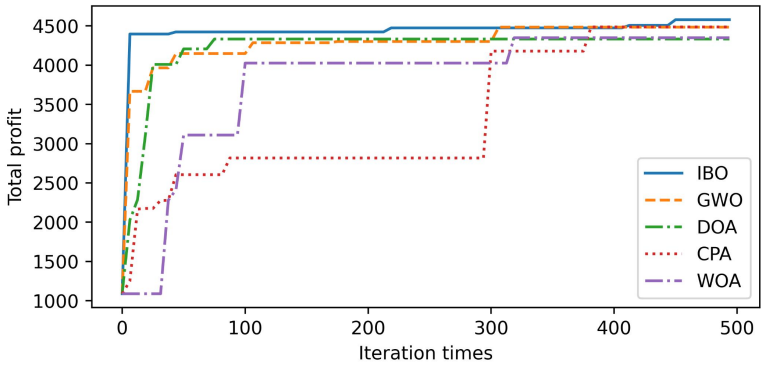

Figure 9. Comparison of IBO algorithm with other solution methods on case 5.

5. Conclusions

This work studies a new linear disassembly line balancing problem arising from manual and human–robot collaborative disassembly lines for e-waste. The focal points encompass hazardous disassembly penalties and switching time, with the overarching objective of maximizing total profit by striking a balance between the recycling revenue, disassembly cost, and hazardous disassembly penalty. A mixed-integer linear program is formulated to model this challenging problem. For small- and medium-scale cases, optimal solutions are efficiently obtained by directly solving the model with CPLEX. However, acknowledging the NP-hard nature of the concerned problem, we design an enhanced brain-storm optimization algorithm. It incorporates problem-specific genetic operators and similarity-based clustering to provide an approximate solution. Experimental results underscore the remarkable performance of the proposed algorithm by comparing it with both competitive peers and solving the proposed mathematical model with CPLEX. The demonstrated high performance of this algorithm suggests its considerable potential for practical industrial applications.

As future research directions, other formal modeling approaches, intelligent optimization algorithms, and data-driven and learning approaches can be explored to solve the proposed problem or further enhance the performance of the presented algorithm. Moreover, exploring the model-solving performance under different configurations of CPLEX, such as inputting an initial feasible solution and selecting different solution strategies, would be meaningful future work.

Author Contributions: Conceptualization, X.G. and Y.T.; methodology, Z.Z.; validation, P.X.; writing—original draft preparation, Z.Z.; writing—review and editing, S.L. and J.W.; visualization, S.Q.; supervision, Y.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant No. 62203093, the Guangdong Basic and Applied Basic Research Foundation under Grant

No. 2021A1515110827, the LiaoNing Revitalization Talents Program under Grant No. XLYC2002041, and Fundamental Research Funds for the Central Universities under Grant No. N2204016.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. 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]
2. 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**, *2023*, 1–12. [CrossRef]
3. Wu, K.; Guo, X.; Zhou, M.; Liu, S.; Qi, L. Multi-objective discrete brainstorming optimizer for stochastic disassembly line balancing problem subject to disassembly failure. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 1224–1229.
4. Li, Z.; Kucukkoc, I.; Zhang, Z. Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem. *Comput. Ind. Eng.* **2019**, *137*, 106056. [CrossRef]
5. 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]
6. Li, J.; Li, J.; Zhang, L.; Sang, H.; Han, Y.; Chen, Q. Solving type-2 fuzzy distributed hybrid flowshop scheduling using an improved brain storm optimization algorithm. *Int. J. Fuzzy Syst.* **2021**, *23*, 1194–1212. [CrossRef]
7. Cervantes-Castillo, A.; Mezura-Montes, E. A modified brain storm optimization algorithm with a special operator to solve constrained optimization problems. *Appl. Intell.* **2020**, *50*, 4145–4161. [CrossRef]
8. Liu, J.; Wang, S. Balancing disassembly line in product recovery to promote the coordinated development of economy and environment. *Sustainability* **2017**, *9*, 309. [CrossRef]
9. Ren, Y.; Zhang, C.; Zhao, F.; Tian, G.; Lin, W.; Meng, L.; Li, H. Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-Optimal algorithm. *J. Clean. Prod.* **2018**, *174*, 1475–1486. [CrossRef]
10. Ilgin, M.A.; Akçay, H.; Araz, C. Disassembly line balancing using linear physical programming. *Int. J. Prod. Res.* **2017**, *55*, 6108–6119. [CrossRef]
11. Zhu, J.F.; Xu, Z.G.; Su, K.Y.; Dong, S.H. Asynchronous parallel disassembly sequence planning for multi-manipulator based on improved shuffled frog leaping algorithm. *SN Appl. Sci.* **2020**, *2*, 1–18. [CrossRef]
12. Liu, J.; Wang, S.W. A Dynamic Coevolution Algorithm for Two-side Sequential Disassembly Line Equilibrium Problem. *J. Manuf. Syst.* **2020**, *29*, 1197–1204.
13. Avikal, S.; Jain, R.; Mishra, P. A heuristic for U-shaped disassembly line balancing problems. *MIT Int. J. Mech. Eng.* **2013**, *3*, 51–56.
14. Salehi, M.; Maleki, H.R.; Niroomand, S. Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms. *Neural Comput. Appl.* **2020**, *32*, 8217–8243. [CrossRef]
15. Özceylan, E.; Paksoy, T. Fuzzy mathematical programming approaches for reverse supply chain optimization with disassembly line balancing problem. *J. Intell. Fuzzy Syst.* **2014**, *26*, 1969–1985. [CrossRef]
16. 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.* **2022**, *19*, 1744–1756. [CrossRef]
17. Zhao, Z.; Liu, S.; Zhou, M.; Guo, X.; Qi, L. Decomposition Method for New Single-Machine Scheduling Problems From Steel Production Systems. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1376–1387. [CrossRef]
18. McGovern, S.M.; Gupta, S.M. A balancing method and genetic algorithm for disassembly line balancing. *Eur. J. Oper. Res.* **2007**, *179*, 692–708. [CrossRef]
19. Kucukkoc, I. Balancing of two-sided disassembly lines: Problem definition, MILP model and genetic algorithm approach. *Comput. Oper. Res.* **2020**, *124*, 105064. [CrossRef]
20. Kalayci, C.B.; Polat, O.; Gupta, S.M. A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Ann. Oper. Res.* **2016**, *242*, 321–354. [CrossRef]
21. Yolmeh, A.; Kianfar, F. An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Comput. Ind. Eng.* **2012**, *62*, 936–945. [CrossRef]
22. Liu, Q.; Li, Y.; Fang, Y.; Laili, Y.; Lou, P.; Pham, D.T. Many-objective best-order-sort genetic algorithm for mixed-model multi-robotic disassembly line balancing. *Procedia CIRP* **2019**, *83*, 14–21. [CrossRef]
23. Zhou, Y.; Li, J.; Hao, J.K.; Glover, F. Detecting critical nodes in sparse graphs via "reduce-solve-combine" memetic search. *Informs J. Comput.* 2023, *ahead of print*.
24. Zhou, Y.; Wang, G.; Hao, J.K.; Geng, N.; Jiang, Z. A fast tri-individual memetic search approach for the distance-based critical node problem. *Eur. J. Oper. Res.* **2023**, *308*, 540–554. [CrossRef]

25. Zhou, Y.; Kou, Y.; Zhou, M. Bilevel memetic search approach to the soft-clustered vehicle routing problem. *Transp. Sci.* **2023**, *57*, 701–716. [CrossRef]
26. Zhou, Y.; Xu, W.; Zhou, M.; Fu, Z.H. Bi-Trajectory Hybrid Search to Solve Bottleneck-Minimized Colored Traveling Salesman Problems. *IEEE Trans. Autom. Sci. Eng.* **2023**, *2023*, 1–11. [CrossRef]
27. 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]
28. 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.* **2020**, *8*, 1199–1209. [CrossRef]
29. Zhao, Z.; Zhou, M.; Liu, S. Iterated Greedy Algorithms for Flow-Shop Scheduling Problems: A Tutorial. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1941–1959. [CrossRef]
30. Zhou, Y.; Xu, W.; Fu, Z.H.; Zhou, M. Multi-neighborhood simulated annealing-based iterated local search for colored traveling salesman problems. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 16072–16082. [CrossRef]
31. Cui, M.; Li, L.; Zhou, M.; Abusorrah, A. Surrogate-assisted autoencoder-embedded evolutionary optimization algorithm to solve high-dimensional expensive problems. *IEEE Trans. Evol. Comput.* **2021**, *26*, 676–689. [CrossRef]
32. Cui, M.; Li, L.; Zhou, M.; Li, J.; Abusorrah, A.; Sedraoui, K. A bi-population cooperative optimization algorithm assisted by an autoencoder for medium-scale expensive problems. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 1952–1966. [CrossRef]
33. Zhu, S.; Xu, L.; Goodman, E.D.; Lu, Z. A new many-objective evolutionary algorithm based on generalized Pareto dominance. *IEEE Trans. Cybern.* **2021**, *52*, 7776–7790. [CrossRef] [PubMed]
34. Zhu, S.; Xu, L.; Goodman, E.D. Hierarchical topology-based cluster representation for scalable evolutionary multiobjective clustering. *IEEE Trans. Cybern.* **2021**, *52*, 9846–9860. [CrossRef] [PubMed]
35. Shi, Y. Brain storm optimization algorithm. In Proceedings of the International Conference in Swarm Intelligence, Chongqing, China, 12–15 June 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 303–309.
36. Bhatt, A.; Dimri, P.; Aggarwal, A. Self-adaptive brainstorming for jobshop scheduling in multicloud environment. *Softw. Pract. Exp.* **2020**, *50*, 1381–1398. [CrossRef]
37. Xuan, H.; Zhao, X.; Liu, Z.; Fan, J.; Li, Y. Energy efficiency opposition-based learning and brain storm optimization for vnf-sc deployment in iot. *Wirel. Commun. Mob. Comput.* **2021**, *2021*. [CrossRef]
38. Li, Z.; Janardhanan, M.N.; Tang, Q. Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots. *Neural Comput. Appl.* **2021**, *33*, 8575–8596. [CrossRef]
39. Xiao, Q.; Guo, X.; Li, D. Partial disassembly line balancing under uncertainty: Robust optimisation models and an improved migrating birds optimisation algorithm. *Int. J. Prod. Res.* **2021**, *59*, 2977–2995. [CrossRef]
40. Xue, J.; Wu, Y.; Shi, Y.; Cheng, S. Brain storm optimization algorithm for multi-objective optimization problems. In Proceedings of the International Conference in Swarm Intelligence, Chongqing, China, 12–15 June 2011; Springer: Berlin/Heidelberg, Germany, 2012; pp. 513–519.
41. Xu, G.; Zhang, Z.; Li, Z.; Guo, X.; Qi, L.; Liu, X. Multi-objective discrete brainstorming optimizer to solve the stochastic multiple-product robotic disassembly line balancing problem subject to disassembly failures. *Mathematics* **2023**, *11*, 1557. [CrossRef]
42. Tîrnăucă, C.; Gómez-Pérez, D.; Balcázar, J.L.; Montaña, J.L. Global optimality in k-means clustering. *Inf. Sci.* **2018**, *439*, 79–94. [CrossRef]
43. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, 1 January 1967; Volume 1, pp. 281–297.
44. 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]
45. 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]
46. 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]
47. Zhang, Z.; Guo, X.; Zhou, M.; Liu, S.; Qi, L. Multi-objective Discrete Grey Wolf Optimizer for Solving Stochastic Multi-objective Disassembly Sequencing and 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. 682–687.
48. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
49. Peraza-Vázquez, H.; Peña-Delgado, A.F.; Echavarría-Castillo, G.; Morales-Cepeda, A.B.; Velasco-Álvarez, J.; Ruiz-Perez, F. A bio-inspired method for engineering design optimization inspired by dingoes hunting strategies. *Math. Probl. Eng.* **2021**, *2021*, 9107547. [CrossRef]
50. Ong, K.M.; Ong, P.; Sia, C.K. A carnivorous plant algorithm for solving global optimization problems. *Appl. Soft Comput.* **2021**, *98*, 106833. [CrossRef]
51. 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]
52. Bao, Z.; Chen, L.; Qiu, K. An aircraft final assembly line balancing problem considering resource constraints and parallel task scheduling. *Comput. Ind. Eng.* **2023**, *182*, 109436. [CrossRef]

53. Kalaycılar, E.G.; Azizoğlu, M.; Yeralan, S. A disassembly line balancing problem with fixed number of workstations. *Eur. J. Oper. Res.* **2016**, *249*, 592–604. [CrossRef]
54. 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]
55. 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]
56. Zitzler, E.; Deb, K.; Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef]