Expanded Discrete Migratory Bird Optimizer for Circular Disassembly Line Balancing with Tool Deterioration and Replacement

Shujin Qin, Feng Guo, Jiacun Wang, Liang Qi, Jiaxin Wang, Xiwang Guo, and Ziyan Zhao

***Abstract*—As an important part of the closed-loop supply chain, the improper disposal of end-of-life (EOL) products reduces overall system efficiency and causes environmental pollution. Thus, it is crucial to dispose of EOL products timely and effectively. Disassembly is a key method for recycling EOL products. During mass disassembly, tools suffer deterioration due to long-term wear and corrosion, increasing processing time for disassembly tasks. To prevent continuous tool deterioration, a tool replacement operation is proposed. Therefore, this work proposes a circular disassembly line balancing with tool replacement, aiming to maximize the disassembly profit by improving the disassembly efficiency as much as possible under the load balance of individual workstations. In this work, a crossover as well as four mutations are provided so as to discretize and extend the original migratory bird optimizer to the solution of the circular disassembly line balancing with tool deterioration and replacement. In the experimental phase, the correctness of the model is verified by the CPLEX solver. The effectiveness of the proposed algorithm is given by comparison experiments due to the CPLEX solver as well as other optimization algorithms. The experimental results show that the proposed algorithm is more effective than the discrete fruit fly optimization algorithm, discrete whale optimization algorithm, and salp swarm algorithm.**

***Note to Practitioners*—This work addresses the improvement of disassembly efficiency and profitability by line design and tool change in the disassembly of end-of-life products. I-shaped line designs that are not fully loaded unbalance the load on the workstations, resulting in some workstations being busy while others are idle. This lengthens workstation processing time and reduces disassembly efficiency. In addition, wear and corrosion of tools used over time increases the processing time of disassembly tasks. To prevent workstation load imbalance and excessive tool aging, we propose a circular disassembly line and a tool replacement strategy. By optimizing the disassembly process, higher efficiency and profitability can be achieved. Experimental results show that our scheme has better load balance and disassembly efficiency. This research is important for practitioners who wish to improve disassembly efficiency and reduce environmental pollution. The method presented in this work is not applicable to the disassembly of large products such as automobiles and airplanes.**

***Key Words*—Circular disassembly line balancing, Tool deterioration, Tool replacement, Migratory bird optimizer.**

I. Introduction

With the rapid advancement of technology, the demand for electromechanical products surges, and consequently, the number of EOL products also rises. Recycling these EOL products using appropriate methods mitigates resource consumption and environmental impact, significantly contributing to sustainable development [1]. Disassembly is a crucial method for recycling EOL products, enabling the recovery of useful subassemblies. In the disassembly line, factors such as the number of tasks per workstation, the wear and tear of disassembly tools, and the execution time and sequence of disassembly tasks affect the efficiency and cost of the process. Balancing these factors to achieve efficient disassembly is known as the disassembly line balancing problem (DLBP) [2].

In the actual disassembly process, many factors affect efficiency [3]. In the factory, disassembly tools separate the required subassemblies from EOL products. However, these tools impact disassembly task processing time due to corrosion, high temperature, or wear from prolonged use, increasing processing time. This situation is called tool deterioration. Tool deterioration is a significant issue in production scheduling [4, 5, 6]. Yang and Kou [7] consider scheduling problems with deteriorating jobs and learning effects, proposing several polynomial-time algorithms to solve single machine scheduling problems. Miao *et al*. [8] consider the parallel machine scheduling problem with step-deteriorating jobs, proposing a polynomial-time optimal algorithm. Wang *et al*. [9] consider resource allocation scheduling with a deterioration effect and position-dependent workloads on a single machine. Previous studies assume the deterioration coefficient of a task is the same on all devices, but in some disassembly processes, it is influenced by the status of the tools in the workstation

Manuscript received March 15, 2025; revised April 3 and April 15, 2025; accepted April 18, 2025.This article was recommended for publication by Associate Editor Yingjun Ji upon evaluation of the reviewers' comments.

Copyright: ©2025 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

This work is supported in part by NSFC under Grant 61903229, in part by Liaoning Revitalization Talents Program under Grant XLYC1907166, in part by the Natural Science Foundation of Shandong Province under Grant ZR2019BF004, and in part by Archival Science and Technology Project of Liaoning Province under Grant 2021-B-004.

S. Qin is with the College of Information and Technology, Shangqiu Normal University, Shangqiu 476000, China (e-mail: sjchin@vip.126.com).

X. Guo, F. Guo and J. Wang are with the College of Artificial Intelligence and Software, Liaoning Petrochemical University, Fushun 113001, China (e-mail: guoxiwang@lnpu.edu.cn, guofeng@stu.lnpu.edu.cn, wangjiaxin@stu.lnpu.edu.cn).

J. Wang is with the Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA (e-mail: jwang@monmouth.edu).

L. Qi is with the Department of Computer Science and Technology at Shandong University of Science and Technology, Qingdao 266590, China (e-mail: qiliang@sdust.edu.cn).

Z. Zhao is with the College of Information Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: neuzhaoziyan@163.com).

Corresponding Author: Shujin Qin.

and the task's complexity. Thus, each task often has varying deterioration coefficients on different workstations.

When the quantity of recycled products is large, prolonged disassembly leads to more severe tool deterioration [10]. To address tool deterioration during large-scale disassembly, this work studies the DLBP considering tool replacement. Long-term use of deteriorating tools seriously affects the efficiency of disassembly tasks, increasing both disassembly time and cost [11]. To mitigate the adverse impact of tool deterioration on disassembly tasks, this work introduces tool replacement operations [12]. When a tool deteriorates significantly, timely tool replacement operations reduce the impact on disassembly efficiency and maximize profits. Additionally, tasks have different deterioration coefficients on different workstations, and the actual processing time of each disassembly task is a linear function of the service time of the corresponding disassembly tool [13].

Since the introduction of the DLBP, research explores DLBP under various disassembly layouts for different problem models and disassembly types [14]. The I-shaped disassembly line has a simple structure, making it easy to understand and implement. Altekin and Tevhide [15] propose a chance-constrained piecewise linear mixed integer programming model. Ilgin *et al*. [16] propose a linear physical programming-based disassembly line balancing method. However, for more complex disassembly types, using I-shaped disassembly lines is less efficient. To address this issue, Hezer and Kara [17] propose an optimization model for parallel DLBP. Agrawal and Tiwari [18] propose the stochastic mixed model U-shaped DLBP. With in-depth research on various DLBPs, researchers continue to explore new disassembly line layouts to better address more complex disassembly problems.

In this work, we investigate tool deterioration and tool replacement during large-scale disassembly processes. Based on the problem characteristics, we propose a new layout for disassembly lines: circular disassembly lines, as shown in Fig. 1. The workstations of the circular disassembly line are arranged in a circular pattern, allowing subassemblies to be distributed circularly to each workstation. Each disassembly task can be executed on the current workstation or the next, reducing deterioration effects in long-term disassembly. This layout improves space utilization, addresses workstation time constraints, and increases disassembly flexibility. Limited research exists on circular disassembly line layouts and deterioration effects in disassembly lines. In summary, we propose a multi-product circular disassembly line balancing with tool deterioration and replacement(MCTDR). We also establish a disassembly line balancing model to maximize disassembly profits.

Scholars adopt various methods to solve the DLBP, primarily categorized into precise methods and approximate methods. Due to the NP-complete nature of DLBP, solving large-scale DLBP using precise methods is challenging. Compared to precise methods, approximate methods obtain optimal or near-optimal solutions within a reasonable time. Approximate methods mainly include heuristic and meta-heuristic methods. Yao and Gupta [19] propose a fish school search algorithm to solve the sequence-dependent DLBP on a U-shaped layout. Zhou *et al*. [20] develop a backward recursive algorithm to solve the DLBP in remanufacturing. Ren *et al*. [21] propose a 2-optimal algorithm to solve the DLBP with multiple objectives. Liu *et al*. [22] propose an improved discrete Bees algorithm to solve the collaborative optimization of robotic disassembly sequence planning and robotic DLBP. Li *et al*. [23] develop a multi-objective immune-mechanism collaborative genetic algorithm based on a Pareto set to solve the multi-objective DLBP. Zhu *et al*. [24] propose a Pareto firefly algorithm to solve the multi-objective DLBP with hazard evaluation.

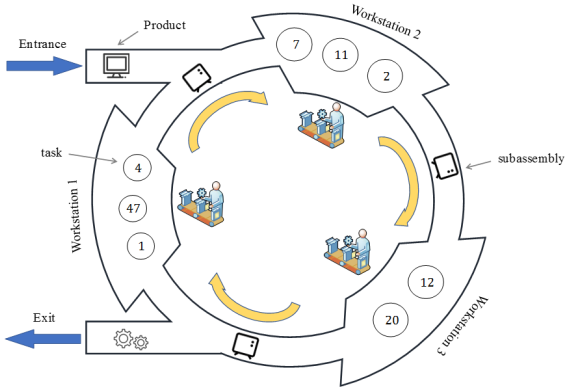


Fig. 1. The circular disassembly line layout.

The Migrating Birds Optimization (MBO) algorithm is a meta-heuristic algorithm that simulates the behavior of migratory birds flying in a V-shaped formation to reduce energy consumption and has advantages such as fewer parameters, ease of understanding, and a simple structure [25]. Alp and Alkaya [26] apply MBO to a fairness-oriented integrated shift scheduling problem. Zhang *et al*. [27] use the MBO algorithm to solve the Type-I multi-manned assembly line balancing problem. Li *et al*. [28] apply MBO to solve the robotic U-shaped assembly line balancing problem. Qin *et al*. [29] use MBO to solve the stochastic DLBP. MBO has been successfully applied to various optimization problems. For the MCTDR in this work, we use expanded discrete migratory bird optimizer (EDMBO) to solve it.

Compared to existing research, this work makes the following contributions:

1) Considering the deterioration of disassembly tools due to wear, high temperature, or corrosion during long-term disassembly, this work provides a functional relationship between disassembly tasks and tool usage time. Considering tool deterioration during large-scale disassembly, this work proposes a tool replacement operation.
2) A new disassembly line layout, circular disassembly line, is designed. Based on the issues raised in this work, a model of MCTDR is established with the goal of maximizing disassembly profits.
3) This work proposes EDMBO to solve the MCTDR. The evolutionary process of migratory birds is designed to enable the algorithm to find the optimal solution faster.

The rest of this article is organized as follows. In the section II, MCTDR is described and a mathematical model

is established. Section III introduces the process of EDMBO algorithm. Section IV presents the comparative experimental results. Section V summarizes the entire text.

II. Problem Description

A. Content Description

This work investigates tool deterioration and disassembly issues in large-scale scenarios. When many products require disassembly, tool deterioration is inevitable. Traditional disassembly line layouts, such as linear, U-shaped, and parallel, are limited by space and workstation cycle time, resulting in tasks not being allocated to suitable workstations, exacerbating the impact of tool deterioration on efficiency. The circular disassembly line layout breaks these limitations. Workstations in the circular disassembly line are arranged in a circular pattern, allowing parts not yet disassembled to move in a loop until a suitable position is found.

In the actual disassembly process, task execution time can increase due to tool wear and worker fatigue, a situation referred to as the deterioration phenomenon. When many EOL products require disassembly, tool wear can exacerbate task deterioration. Therefore, this work introduces a tool replacement operation to replace worn tools at appropriate times, reducing the impact of tool wear on task execution time and increasing profits.

To represent the tool replacement operation in the disassembly scheme, this work establishes a set of virtual tasks corresponding to the types of tools required for disassembly. Each virtual task corresponds to a type of disassembly tool. Executing a virtual task equates to performing a tool replacement operation.

Fig. 2 presents a schematic diagram of the tool replacement operation for a refrigerator. The blue tasks represent normal disassembly tasks, the orange tasks represent virtual tasks, and the number on each task indicates the disassembly tool used. Two normal disassembly tasks on workstation 2 require the use of tool 1. Due to tool deterioration, using tool 1 for the second normal disassembly task extends disassembly time, increasing costs. To reduce the impact of tool deterioration on efficiency, this work adds a virtual task for replacing tool 1 between these two normal disassembly tasks, promptly replacing tool 1.

B. Disassembly Precedence Graph

The disassembly precedence graph is an invaluable tool in product design and engineering. It identifies the most efficient sequence for dismantling a product. Analyzing the graph determines the order in which subassemblies should be removed, enabling the effective disassembly of EOL products. Removing subassemblies in an optimal sequence reduces the likelihood of damage to individual subassemblies and facilitates the identification and separation of reusable components. Therefore, this work uses the disassembly precedence graph to describe the disassembly information of the product.

The disassembly precedence graph typically consists of nodes representing disassembly tasks and directed edges indicating the sequence in which these tasks should be performed.

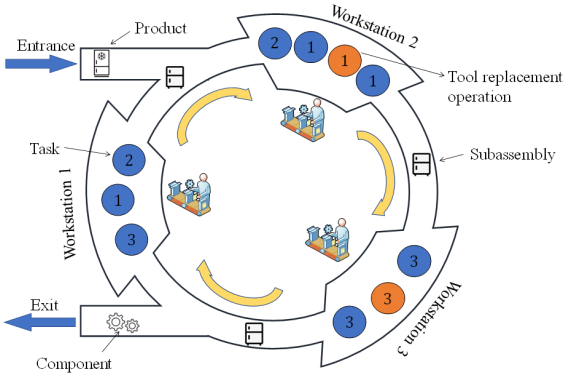


Fig. 2. Refrigerator tool replacement operation schematic.

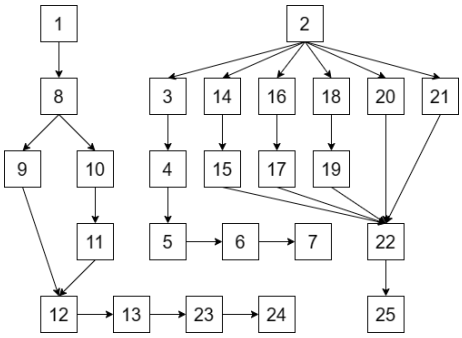


Fig. 3. Disassembly precedence graph of the refrigerator.

Following the directed edges in the graph allows the correct and efficient disassembly of required subassemblies.

For example, Fig. 3 depicts the disassembly precedence graph of a refrigerator. Squares represent disassembly tasks, and directed edges indicate their disassembly priority. The graph shows that task 1 must precede task 8. Task 8 must be completed before executing task 9 or task 10.

In order to store the product information into the computer, we design a precedence matrix $P = [p_{ij}^g]$ and a resource association matrix $D = [d_{ir}^g]$.

1) Precedence matrix

The precedence matrix $P = [p_{ij}^g]$ is used to describe the precedence relation between the current disassembly task i and other disassembly tasks j in product g. It is defined as:

$$p_{ij}^g = \begin{cases} 1, & \text{if task } i \text{ is executed before task } j \text{ in product } g; \\ 0, & \text{otherwise.} \end{cases}$$

The disassembly precedence graph of the product indicates the precedence relationship between disassembly tasks, which can be represented using a precedence matrix. For example, Fig. 3 illustrates:

$[p_{11}^1]$=0, $[p_{18}^1]$=1, $[p_{89}^1]$=1.

2) Resource association matrix

Resource association matrix $D = [d_{ir}^g]$ is used to describe the use relationship between disassembly task i and disassembly tool r in product g, which is defined as:

$$d_{ir}^{g}=\begin{cases}1, & \text{if tool } r \text{ is required to perform task } i \text{ in product } g;\\ 0, & \text{otherwise.}\end{cases}$$

Each workstation in the disassembly line is equipped with various numbered tools required for product disassembly. The resource association matrix identifies the relationship between tasks and tools when a task is assigned to a workstation, allowing the corresponding tools to perform the disassembly task. Table I illustrates the relationship between tasks and tools for disassembling the refrigerator. From Table I, it is evident that tool 3 is required for task 1, and tool 1 is required for task 2.

In order to establish a single objective circular disassembly line model, we make the following assumptions:

1) Matrices P and D are known.
2) The deterioration coefficient of each disassembly tool is known.
3) The number of workstations is limited.
4) The open workstation is assigned at least one disassembly task.
5) The time spent performing the disassembly task is linearly related to the use time of the disassembly tool.
6) The deterioration cost per unit time and the normal execution time and execution cost of each disassembly task are known.

C. Mathematical Model

We establish a mathematical model for the problem proposed in this article. The notations and decision variables in the mathematical model are defined as follows:

Sets:

- $\mathbb{G}$ set of products,$\mathbb{G}=\{1,2,\ldots,G\}$, where G is the number of products.
- $\mathbb{W}$ set of workstations,$\mathbb{W}=\{1,2,\ldots,W\}$, where W is the number of workstations.
- $\mathbb{K}$ set of locations,$\mathbb{K}=\{1,2,\ldots,K\}$, where K is the number of locations on the workstation.
- $\mathbb{R}$ set of disassembly tools,$\mathbb{R}=\{1,2,\ldots,R\}$, where R is the number of tools.
- $\mathbb{I}_g$ the set of tasks in product g.
- $\mathbb{I}_{g,i}^{con}$ set of tasks that conflict with task i in product g.
- $\mathbb{I}_{g,i}^{pre}$ immediately preceding task set for task i in product g.

TABLE I Relationship between Tasks and Tools.

Task index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
Tool number	3	1	2	3	2	1	2	2	3	1	2	2	3	1	...

Parameters:

- i, j task indexes.
- g product index,$g \in \mathbb{G}$.
- w workstation index,$w \in \mathbb{W}$.
- k location index,$k \in \mathbb{K}$.
- r disassembly tool index,$r \in \mathbb{R}$.
- $T_{w,i}^{g}$ normal disassembly time of task i in product g on workstation w.
- $\alpha_{w,i}^{g}$ deterioration coefficient of processing task i in product g on workstation w.
- V_i^g the value of performing the i-th task of product g.
- c deterioration cost per unit of time.
- c_i^g disassembly cost of task i in product g.
- c^W fixed cost of opening the workstation.
- R_i^g attribute of task i in product g.

Decision variables:

- $S_{w,i,r}^{g}$ the time that tool r on the w-th workstation has been used before task i in product g is executed.
- $T_{i,w,k}^{g}$ the start time of task i in product g at the k-th location on the w-th workstation.
- $T_{g,w,i}^{'}$ actual disassembly time of task i in product g on the w-th workstation.
- $T_{g,w,i}^{D}$ deterioration time of task i in product g on the w-th workstation.

$$x_{i,w,k}^{g}=\begin{cases}1, & \text{if task } i \text{ in product } g \text{ is executed at } k\text{-th}\\ & \text{position on workstation } w;\\ 0, & \text{otherwise.}\end{cases}$$

$$u_w=\begin{cases}1, & \text{if workstation } w \text{ is used;}\\ 0, & \text{otherwise.}\end{cases}$$

The following is a mathematical model to describe the problem considered in this work.

$$max \sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} V_i^g x_{i,w,k}^g - \sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} c_i^g x_{i,w,k}^g - \sum_{w\in\mathbb{W}} c^W u_w - \sum_{g\in\mathbb{G}}\sum_{w\in\mathbb{W}}\sum_{i\in\mathbb{I}_g} cT_{g,w,i}^{D} \tag{1}$$

$$\sum_{g\in\mathbb{G}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} x_{i,w,k}^{g} \le 1, \forall i \in \mathbb{I}_g \tag{2}$$

$$u_w \ge u_{w+1}, \forall w \in \mathbb{W} \tag{3}$$

$$\sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g} x_{i,w,k}^{g} \le 1, \forall w \in \mathbb{W}, k \in \mathbb{K} \tag{4}$$

$$\sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g} x_{i,w,k}^{g} \ge \sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g} x_{i,w,k+1}^{g}, \forall w \in \mathbb{W}, k \in \mathbb{K} \tag{5}$$

$$\sum_{g\in\mathbb{G}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} x^g_{i,w,k} + \sum_{g\in\mathbb{G}}\sum_{i'\in\mathbb{I}^{con}_{g,i}}\sum_{w\in\mathbb{W}}\sum_{k\in\mathbb{K}} x^g_{i',w,k} \le 1, \quad \forall i \in \mathbb{I}_g \tag{6}$$

$$\sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g}\sum_{k\in\mathbb{K}} x^g_{i,w,k} \ge u_w, \forall w \in \mathbb{W} \tag{7}$$

$$\sum_{g\in\mathbb{G}}\sum_{i\in\mathbb{I}_g}\sum_{k\in\mathbb{K}} x^g_{i,w,k} \le M u_w, \forall w \in \mathbb{W} \tag{8}$$

$$x^g_{i,w,k} \le \sum_{g\in\mathbb{G}}\sum_{i'\in\mathbb{I}^{pre}_{g,i}}\sum_{w'\in\mathbb{W}}\sum_{k'\in\mathbb{K}} x^g_{i',w',k'}, \quad \forall i \in \mathbb{I}_g, w \in \mathbb{W}, k \in \mathbb{K} \tag{9}$$

$$T^g_{i,w,k+1} \ge T^g_{i',w,k} + T'_{g,w,i'}, \quad \forall g \in \mathbb{G}, i, i' \in \mathbb{I}_g, w \in \mathbb{W}, k \in \mathbb{K} \tag{10}$$

$$T^g_{i,w,k} \ge T^g_{i',w',k'} + T'_{g,w',i'} + M\left(x^g_{i,w,k} + x^g_{i',w',k'} - 2\right), \quad \forall g \in \mathbb{G}, i \in \mathbb{I}_g, w, w' \in \mathbb{W}, k, k' \in \mathbb{K}, i' \in \mathbb{I}^{pre}_{g,i} \tag{11}$$

$$S^g_{w,i,r} \ge S^g_{w,j,r} + T'_{g,w,j} d^g_{j,r} + M\left(x^g_{i,w,k} + x^g_{j,w,k-1} - 2\right) - M\left(R^g_i d^g_{i,r} + R^g_j d^g_{j,r}\right), \quad \forall g \in \mathbb{G}, i, j \in \mathbb{I}_g, w \in \mathbb{W}, r \in \mathbb{R}, k \in \mathbb{K} \tag{12}$$

$$T'_{g,w,i} \ge T^g_{w,i} + T^D_{g,w,i} + M\left(x^g_{i,w,k} - 1\right), \quad \forall g \in \mathbb{G}, i \in \mathbb{I}_g, w \in \mathbb{W}, k \in \mathbb{K} \tag{13}$$

$$T^D_{g,w,i} \ge d^g_{i,r}\alpha^g_{w,i} S^g_{w,i,r}, \forall g \in \mathbb{G}, i \in \mathbb{I}_g, w \in \mathbb{W}, r \in \mathbb{R} \tag{14}$$

Objective function (1) maximizes disassembly profits. Constraint (2) ensures each task is executed at most once. Constraint (3) ensures workstations are activated in sequence. Constraint (4) ensures at most one task is performed at each location. Constraint (5) indicates positions in the workstation must be assigned tasks in sequence. Constraint (6) indicates for conflicting disassembly tasks, at most one can be performed. Constraints (7) and (8) ensure open workstations must assign tasks, and unopened workstations cannot assign tasks. Constraint (9) ensures the task precedence relationship is satisfied between disassembly tasks. Constraint (10) ensures the task at the next position cannot be executed until the task at the previous position is completed. Constraint (11) ensures the immediate predecessor of task i is executed before task i. Constraint (12) represents the time each tool is used before disassembly task i is performed. Constraint (13) represents the actual disassembly time of task i, which equals the normal disassembly time of task i plus its deterioration time at the workstation. Constraint (14) represents the deterioration time of the task at the workstation.

III. Proposed Algorithm

A. Expanded Discrete Migratory Bird Optimizer

MBO originates from observing and researching migratory birds' behavior. MBO simulates various behaviors of birds during migration, including flight patterns, environmental adaptability, and interactive behaviors. The basic idea of this algorithm views the search space as a set of potential solutions for optimization problems and uses a strategy to simulate the selection and interaction behavior of migratory birds during migration to find the optimal solution. Specifically, MBO involves three key steps: population initialization, population update, and leader bird replacement. Population initialization and leader bird replacement construct the initial and new generation solution sets. Population update evaluates and sorts solutions in the current population, adjusting the search strategy accordingly.

The main feature of MBO is neighborhood solution sharing, a strategy introduced during the population update stage. It allows individuals to exchange information within their neighborhoods, thereby improving search efficiency and quality. Neighborhood solution sharing is an effective mechanism. It helps algorithms escape local optima and explore a wider range of regions in the search space. It reduces redundant operations in the search space, making the algorithm more efficient and stable. Neighborhood solution sharing is an essential component of MBO algorithms, producing significant optimization effects in various practical applications.

Compared to other popular optimization algorithms, MBO exhibits a higher rate of convergence and superior global optimization capability. Therefore, we select EDMBO, after discretizing and optimizing MBO, to solve the MCTDR problem presented in this work.

In EDMBO, to clearly describe the problem, we design a two-stage encoding method to represent the solution. We design a crossover operator and four mutation strategies to guide the population evolution in the desired direction, helping migratory birds generate neighborhood solutions. To increase population diversity, we conduct mutation operations with follower birds to further explore better solutions. The detailed steps of EDMBO are shown in Algorithm 1.

B. Encoding and Decoding

When solving the DLBP, encoding and decoding are crucial concepts. For the DLBP considering tool replacement in this work, we aim to obtain the optimal solution that includes the execution sequence of tasks and their allocation on workstations. Therefore, we design a binary encoding X (x^1, x^2) to represent the disassembly plan, where x^1 is the task sequence and x^2 is the workstation assignment for each task. As shown in Fig. 4, the task sequence of this disassembly plan is x^1 = (1, 47, 4, 7, 11, 2, 12, 20), and the workstation sequence is x^2 = (1, 1, 1, 1, 2, 2, 2). Tasks 1, 47, 4, 7, and 11 are assigned to workstation 1, while tasks 2, 12, and 20 are assigned to workstation 2.

Decoding is the process of converting encoded solutions into practical solutions. Based on the algorithm, we can derive the disassembly task sequence and workstation sequence to

Algorithm 1 Expanded Discrete migratory bird optimizer

Input: population size, number of iterations
Output: the best solution set X
Begin
Initialize population.
Construct a V formation queue.
while (g <maximum number of iterations) **do**
for ($k = 0$; k <population size; $k++$) **do**
Individual evolution.
end for
Replacement of the leader bird.
$g = g + 1$
Update X.
end while
return X
End

determine the disassembly profit of the solution. Encoding and decoding techniques are particularly important in solving DLBP, as they help algorithms better adapt to the problem's characteristics, reduce computational complexity, and improve efficiency and accuracy.

The abstract encoding scheme becomes concrete through the following worked example, which also illustrates how precedence constraints are enforced during solution decoding:

To concretize the encoding/decoding process, consider a refrigerator disassembly instance with the following encoded solution:

$$x^1 = (2, 5, 1, 8, 14) \quad \text{(Task sequence)}$$
$$x^2 = (1, 1, 2, 2, 3) \quad \text{(Workstation assignments)}$$

Step 1: Decoding to Disassembly Plan

1) *Task Allocation*:
   - Workstation 1: Tasks 2, 5 (Tools: 1, 2)
   - Workstation 2: Tasks 1, 8 (Tools: 3, 2)
   - Workstation 3: Task 14 (Tool: 1)
2) *Precedence Validation*: Check against Fig. 3's precedence matrix P:
   - Task 1 $\prec$ Task 8 (satisfied)
   - Task 2 $\prec$ Task 5 (violated, triggers repair in Step 2)

Step 2: Priority Repair During Crossover When precedence violations occur (e.g., Task 5 before Task 2), the algorithm:

- Identifies conflict via $p^{pre}_{g,5} = \{2\}$ (from Section II.B)
- Swaps positions to get valid sequence: $(2, 5, 1, 8, 14) \rightarrow (2, 1, 5, 8, 14)$

Step 3: Profit Computation Using Eq. (1) with parameters from Table I: This example demonstrates how encoded solutions map to executable plans while maintaining precedence constraints.

C. Population Initialization and Construction of V-shaped Queues

The initialization of the population in EDMBO significantly impacts the search ability and final results of the algorithm.

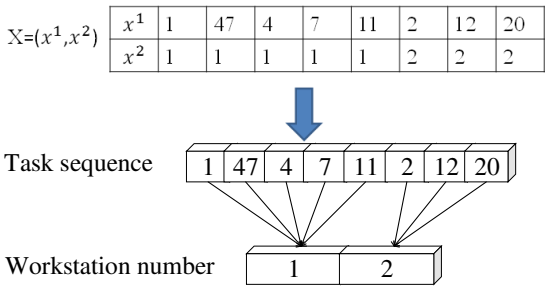


Fig. 4. Encoding structure.

This work uses a method of randomly generating a certain number of solutions for the initial population, ensuring these solutions are uniformly distributed in the solution space.

According to EDMBO characteristics, after generating a population, individuals need sorting to construct a V-shaped queue. First, sort individuals based on target value, then select the best individual as the leader bird. The remaining individuals are placed in left and right queues according to their target values, forming a V-shape. The constructed V-shaped queue helps migratory bird populations explore the search space and find the optimal solution globally, improving EDMBO's search efficiency and accuracy.

D. Individual Evolution

In EDMBO, leader and follower birds evolve individually using neighborhood solutions. A neighborhood solution is a candidate solution that explores the surrounding environment. By comparing the target value of the neighborhood solution with the current optimal solution, it is determined whether the migratory bird is replaced or maintains its original position.

The leader bird generates its neighborhood solution through a crossover operation with the left and right follower birds, then performs a neighborhood search. If a better solution is found in the neighborhood, the leader bird is replaced with the optimal neighborhood solution. After the search, unused neighborhood solutions of the leader bird are transferred to the left and right follower birds.

The left and right migratory birds generate neighborhood solutions through crossover with subsequent birds and perform a neighborhood search between their solutions and those not used by the previous bird. If a bird finds a better solution than its current one, it replaces its current solution with the optimal one. After the search, unused solutions of the current bird are transferred to the next bird.

The steps of the crossover operation are shown in Fig. 5. First, select the current bird and its subsequent bird as parent 1 and parent 2, respectively. Then, randomly generate a binary code equal in length to the disassembly sequence. Parents 1 and 2 generate new individuals based on the binary code. In binary encoding, 0 represents selecting the task from the current bird, while 1 represents selecting the task from the subsequent bird.

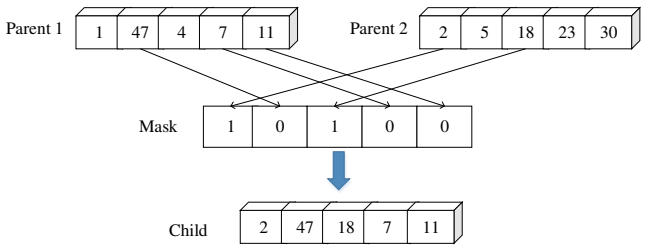


Fig. 5. Process of crossover.

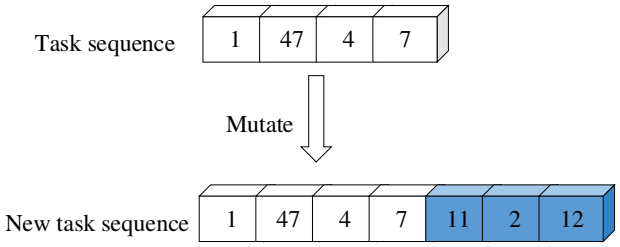


Fig. 6. The first mutation strategy.

To facilitate neighborhood solutions in aiding individual evolution, we select high-quality individuals as mutation candidates within a neighborhood. We design four mutation strategies. When generating neighborhood solutions, we randomly select one to use for new individuals. The four mutation strategies are as follows:

1) As shown in Fig. 6, new individuals generated through crossover operations sometimes have shorter disassembly task sequences. To address this, we randomly add tasks to the current sequence, ensuring the total number of tasks is not exceeded.
2) As shown in Fig. 7, the order of disassembly tasks in a sequence can impact the disassembly results. We randomly select a disassembly task from the sequence of the current individual, and then insert it into a new position while ensuring that the disassembly precedence relationship is maintained.
3) As illustrated in Fig. 8, during the disassembly process, the performance of the disassembly tool gradually deteriorates. To mitigate the impact of tool wear on task disassembly time, tools at the workstation need timely replacement. Given the potential need for multiple tool replacements, we randomly insert a tool replacement operation into the current task sequence.
4) Similarly, as depicted in Fig. 9, to prevent arbitrary tool replacements during disassembly, we randomly eliminate a tool replacement operation from the current task sequence.

E. Replacement of Leader Bird

After a specified number of evolutions for leader and follower birds, the initial population and all new individuals are assembled into a new set. To enhance population diversity,

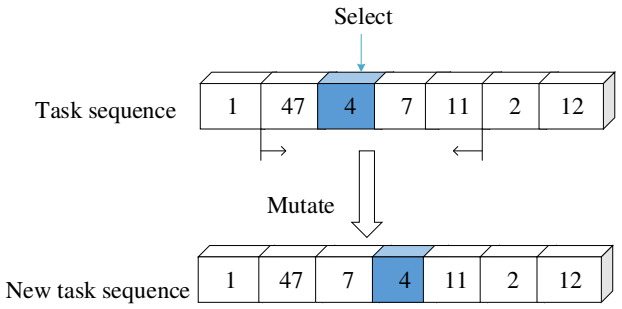


Fig. 7. The second mutation strategy.

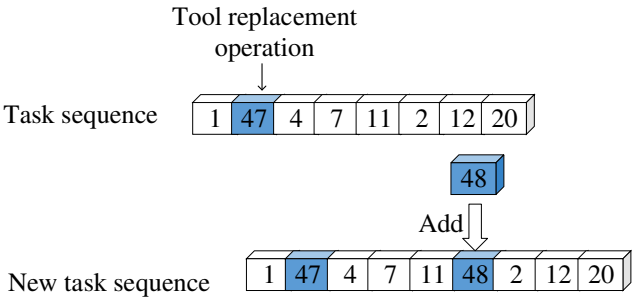


Fig. 8. The third mutation strategy.

follower birds from the initial population undergo mutation and are incorporated into this new set. Subsequently, individuals in the new set are sorted, the top n optimal individuals are selected to form a new generation population, and a V-shaped queue is constructed.

To quantify the individual contributions of each mutation strategy, we disable one strategy at a time while keeping the others active, using Case 5 as a baseline. The results are shown in Table II.

The most pronounced decrease in profit (7.7%) was observed when mutations3 were disabled, validating their key role in reducing the cost of deterioration. In contrast, removing mutation 4 has a minimal impact (2.3% decrease), suggesting that its primary function is to preserve diversity rather than di-

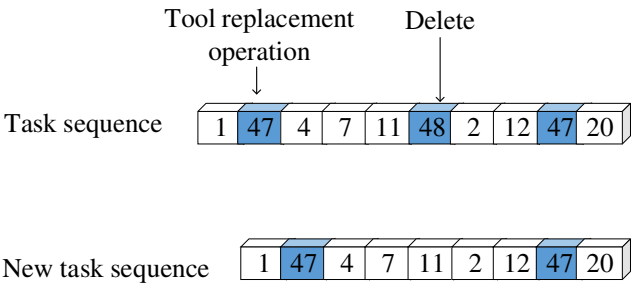


Fig. 9. The fourth mutation strategy.

TABLE II Performance impact of removing individual mutation strategies

Mutation strategies	Profit	Time (s)	Workstations	Tool Replacement
No remove	1373	42	4	3
1	1295 (-5.7%)	51 (+21%)	5	4
2	1318 (-4.0%)	47 (+12%)	4	3
3	1267 (-7.7%)	39 (-7%)	5	**0**
4	1342 (-2.3%)	44 (+5%)	4	4

rectly optimize profits. Interestingly, while disabling mutation 1 increases the runtime by 21%, it also reduces the quality of the solution, suggesting that this strategy has the dual role of accelerating convergence and getting rid of local optima. These findings justify the inclusion of all four strategies, even though their activation probabilities are based on the adaptive activation probabilities of the problem phases.

IV. Experimental Studies

A. Test Instances

TABLE III Product Set

Product	Num of tasks	Num of required tools
Washing machine	13	3
Refrigerator	25	3
Radio	30	3
Hammer drill	46	3

TABLE IV Case Information

Case	Product				Num of tasks
	Washing machine	Refrigerator	Radio	Hammer drill	
1	1	0	0	0	13
2	0	1	0	0	25
3	2	0	0	0	26
4	0	0	1	0	30
5	0	0	0	1	46
6	0	0	2	0	60
7	0	0	1	1	76

We use the IBM ILOG CPLEX solver to validate the mathematical model and record the optimal solutions for each experimental case. Then, we use EDMBO to solve the same cases and compare the solutions obtained by CPLEX and EDMBO. The operating environment is Windows 10, with an AMD A6-9210 Radeon R4, 5 Compute Cores 2C+3G 2.40GHz CPU.

We select the washing machine, refrigerator, radio, and hammer drill as the products for disassembly. Given the focus on a multi-product DLBP in this study, these four products are paired to create various experimental scenarios for solving the problem. Table III details the specific information for each product, while Table IV presents the combinations of experimental cases.

B. Analysis of Experimental Results

Circular disassembly lines can be used in a wide range of disassembly scenarios. In this study, the radio is used as the product and the number of workstations is 3. Fig. 10 visually shows the results of the experiment. With the same number of stations and other parameters, circular disassembly lines are no less efficient in terms of utilization and have an advantage in terms of workstation load balance compared to I-shaped and U-shaped disassembly lines, regardless of whether the disassembly line is fully loaded or not, and regardless of whether the product is complex or not. Load balancing prevents overuse of the same tool, which slows the rate of tool deterioration.

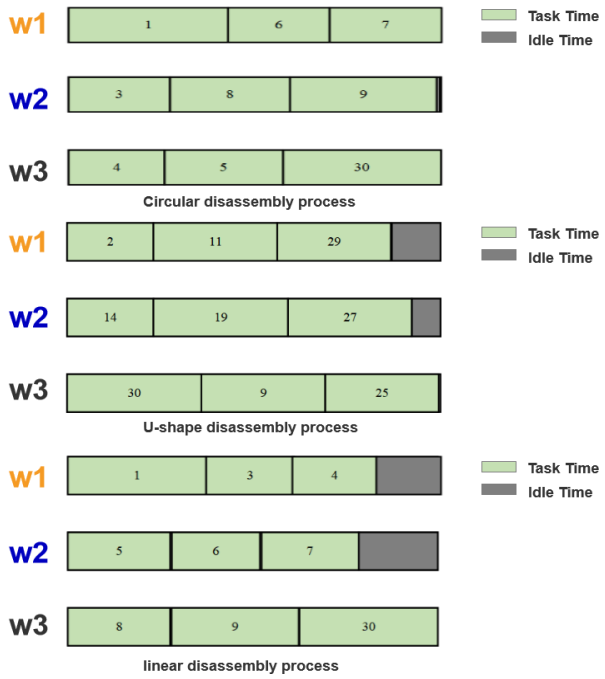


Fig. 10. Relative load of workstations in different line types

Fig. 10 provides a quantitative comparison of workstation load characteristics for different disassembly line layouts. Fig. 10 shows three main patterns: the I-shaped disassembly line exhibits a significant load imbalance, with workstation 3 consistently having a higher workload than workstation 1 due to the sequential nature of the task assignments. the distribution of the U-shaped disassembly line layout improves, but is still uneven. The circular disassembly line layout shows nearly uniform utilization. This balance between efficiency and robustness directly supports the tool degradation mitigation strategy in Section IV-C.

We use CPLEX and EDMBO separately to solve each case, with the maximum running time of CPLEX set to 4 hours. The solution results obtained using CPLEX are presented in Table V. For example, considering case 1 as detailed in Table III, it involves 13 disassembly tasks and requires 3 types of tools. To accommodate these tool types, we introduce corresponding virtual tasks: task 14 for replacing tool 1, task 15 for replacing tool 2, and task 16 for replacing tool 3. In case 1, tasks 1, 4, ..., and 13 are assigned to workstation 1. Task 14 appears twice, indicating two instances of tool 1 replacement. The term ”Profit” denotes the proceeds generated from the disassembly sequence, while ”Time” refers to the computational

time required by CPLEX to derive the sequence.

TABLE V CPLEX Solutions

Case	Disassembly sequence	Profit	Solution status	Time(s)
1	(1, 4, 14, 9, 14, 13)	1167	optimal	132.93
2	-	-	-	14400
3	-	-	-	14400
4	(31, 1, 31, 31, 31, 31, 4, 16) → (30) → (32, 28) → (31) → (3, 33, 5, 33, 29, 32, 32, 32, 33, 9)	526	feasible	14400
5	-	-	-	14400
6	-	-	-	14400
7	-	-	-	14400

For each case, the solution obtained by CPLEX is categorized into two states. "Optimal" indicates that the disassembly sequence obtained is the optimal solution at that time. "Feasible" indicates that the disassembly sequence obtained is not optimal but remains feasible within the maximum running time. The results from applying CPLEX to solve various cases highlight the high complexity of the model proposed in this work. For small-scale cases, CPLEX can either find an optimal solution or a feasible solution within the specified time limit. However, as the case scale increases, CPLEX may struggle to find even a feasible solution within the given time constraints.

The solution results based on EDMBO are presented in Table VI. Compared to CPLEX, EDMBO consistently provides feasible solutions across cases of varying scales. In instances where CPLEX finds the optimal solution, EDMBO achieves comparable results in a shorter time. Moreover, for cases where CPLEX yields feasible solutions or fails to find any, EDMBO consistently delivers higher quality solutions within shorter computational periods.

TABLE VI EDMBO Solutions

Case	Disassembly sequence	Profit	Time(s)
1	(1, 4, 14, 9, 14, 13)	1167	0.255
2	(1, 2) → (14, 8) → (3, 10, 11) → (9, 4, 18) → (12, 20, 13, 5) → (23, 19) → (21, 16, 17) → (15, 22) → (6, 7, 26, 25, 24)	514.5	13.102
3	(1, 27, 14, 27, 4, 9, 27, 16, 27, 19, 27, 24, 13)	2340	4.396
4	(2, 11, 33, 12, 31, 5, 16, 33, 30) → (28, 29, 9)	576	7.289
5	(1, 47, 4, 7, 11) → (2, 12, 20) → (21, 18, 49, 30) → (31, 13, 22) → (37, 28) → (42, 46, 32) → (38, 23, 43)	1373	11.241
6	(1, 40, 34) → (3, 4, 5) → (6, 61, 7, 35, 8) → (62, 46, 60, 63, 9, 30) → (58, 59, 39)	1153	14.117
7	(31, 32, 11) → (34, 37, 41) → (14, 19, 25) → (42, 51) → (27) → (50, 79, 48, 61, 43) → (29, 52, 78) → (60, 58) → (67) → (9, 62, 77) → (68, 72, 53) → (73, 30, 76)	1933	16.873

The essence of this study involves integrating tool replacement operations into disassembly task assignments. Using case 4 as a focal point, we investigate task allocation scenarios with and without tool replacement operations, with results depicted in Fig. 11. A comparison reveals that incorporating tool replacement operations mitigates the impact of tool deterioration, leading to fewer active workstations and enhanced disassembly profits.

C. Analysis of EDMBO Performance

To evaluate EDMBO's performance in solving MCTDR, we compare it with FOA [30], WOA [31], and SSA [32] using independent experiments on our selected test cases. Each algorithm undergoes 20 experiments where we record the optimal values per generation. To visually assess convergence rates and solution quality, we compute the average optimal value across these experiments and plot the iteration process for each algorithm in Fig. 12. The horizontal axis denotes the number of iterations, while the vertical axis represents the optimal target value per generation. From the iteration curves shown in Fig. 12, it is evident that EDMBO achieves faster convergence rates and consistently delivers higher-quality solutions compared to FOA, WOA, and SSA.

The computational efficiency of EDMBO is also reflected in its resource consumption. This efficiency stems from two architectural choices; (1) the shared neighborhood solution pool reduces redundant population storage, and (2) discrete coding has little memory difference compared to continuous space algorithms such as WOA. This also allows EDMBO to handle more task instances without failures, which is a crucial advantage for real-world disassembly scheduling, since industrial controllers typically have limited RAM. It is worth noting that EDMBO optimizes the memory time tradeoffs and reduces memory usage by 15% compared to SSA, but the convergence speed does not suffer as a result.

In addition to evaluating the algorithm's convergence rate and solution quality, we also assess its stability. We record the optimal, worst, and average values obtained from 20 experiments for each algorithm, as detailed in Table VII. The table shows that EDMBO consistently achieves better solution quality, with higher worst and average values compared to other algorithms. This comparison across different metrics indicates that EDMBO demonstrates robust stability in tackling problems. In summary, EDMBO proves to be well-suited for addressing MCTDR.

As shown in Table VIII, incorporating tool replacement operations reduces active workstations from 5 to 3 and increases profit by 18%. The deterioration time of tools decreases by 37.5%, validating that timely replacement mitigates efficiency loss caused by wear. Fig. 11 further illustrates the task assignment: replacing Tool 1 between Tasks 2 and 12 resets its service time $S^{g}_{w,i,r}$, preventing cumulative deterioration effects.

V. Conclusion

This article proposes a circular disassembly line balancing model that incorporates tool replacement and addresses the impact of disassembly tool deterioration on efficiency and cost. The model aims to maximize disassembly profits while considering both the deterioration and replacement costs of disassembly tools. To handle this, we introduce virtual tasks, whose number corresponds to the types of tools needed for product disassembly; executing a virtual task signifies replacing the associated tool type. To validate the proposed model, we utilize CPLEX to ensure its correctness. For solving the MCTDR, we introduce enhanced dynamic migratory

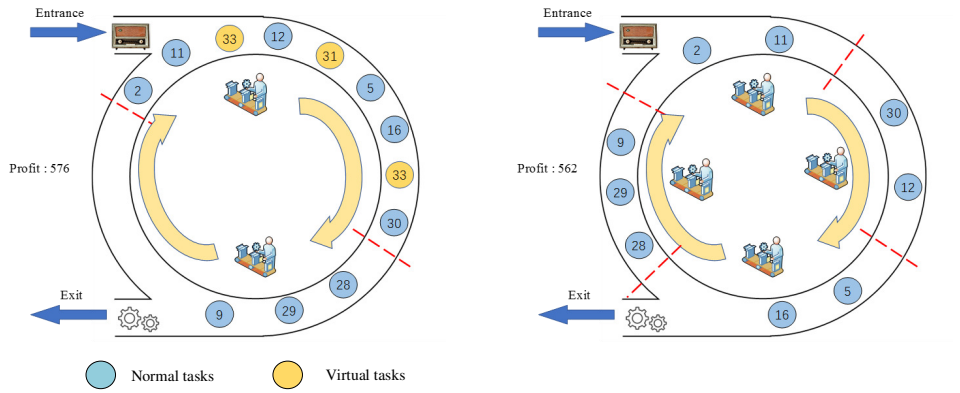
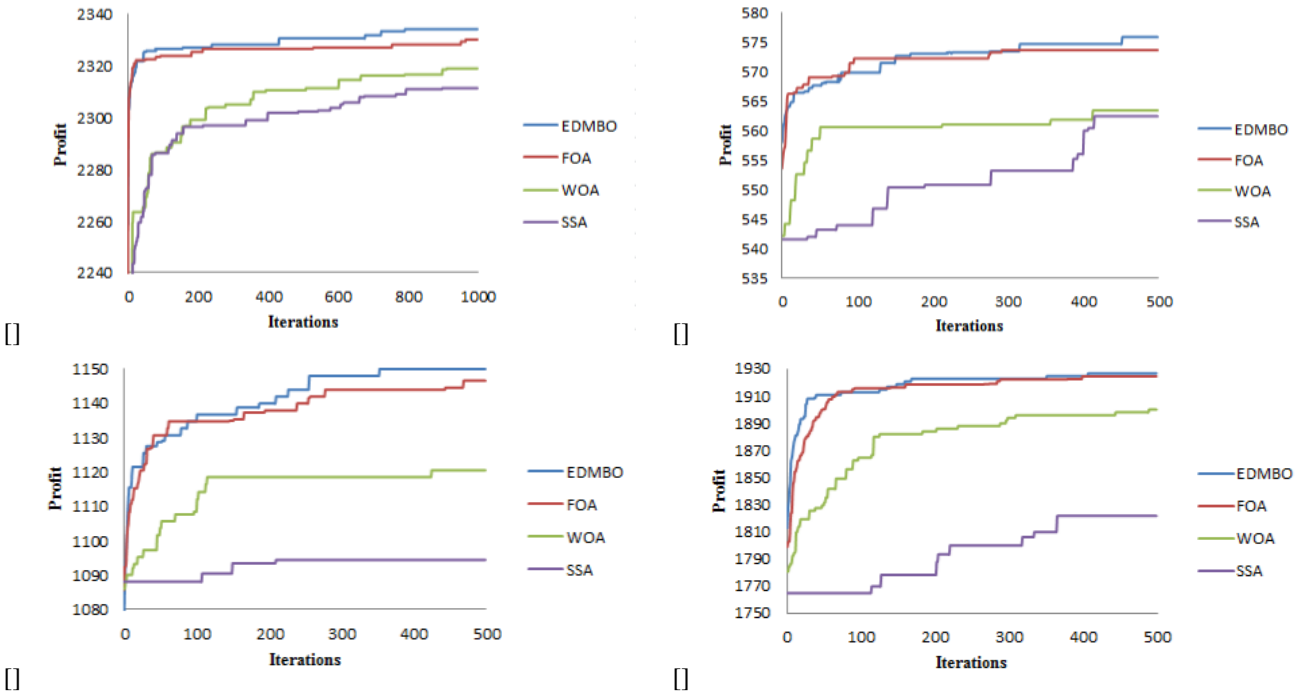


Fig. 11. Assignment of tasks for with and without tool replacement operations of Case 4.



[]

[]

[]

[]

Fig. 12. Iterative comparison. (a) Case 3. (b) Case 4. (c) Case 6. (d) Case 7.

TABLE VII Experimental Results of Stability Test.

Case	Optimal				Worst				Average			
	EDMBO	FOA	SSA	WOA	EDMBO	FOA	SSA	WOA	EDMBO	FOA	SSA	WOA
1	1167	1167	1167	1167	1147	1137	1069	1075	1157	1152	1118	1121
2	514.5	515	465	499	429	410	369	393.5	471.75	462.5	417	446.25
3	2340	2340	2312.5	2324	2228	2197	2112	2105	2284	2268.5	2212.25	2214.5
4	576	576	568	566	554	537	531	533	565	556.5	549.5	549.5
5	1373	1363	1318	1333	1267.5	1272	1262	1231	1320.25	1317.5	1290	1282
6	1153	1150	1099	1131	1062	1061	1042	1038	1107.5	1105.5	1070.5	1084.5
7	1933	1928	1847	1916	1764.5	1768	1729	1715.5	1848.75	1848	1788	1815.75

TABLE VIII Comparison of Disassembly With and Without Tool Replacement Operations (Case 4).

Metric	Without Replacement	With Replacement	Improvement
Active Workstations	5	3	40% ↓
Disassembly Profit	526	576	9.5% ↑
Tool Deterioration Time	120	75	37.5% ↓
Workstation Utilization	68%	89%	21% ↑

bird optimization. The effectiveness of EDMBO in solving MCTDR is demonstrated by conducting experiments on three different scales, large, medium and small, comparing results with other intelligent optimization algorithms.

Our next step involves integrating additional optimization objectives within circular disassembly lines and delving deeper into challenges associated with tool deterioration. Furthermore, we aim to explore advanced intelligent optimization algorithms tailored for addressing DLBP.

REFERENCES

[1] S. Qin, S. Zhang, J. Wang, S. Liu, X. Guo, and L. Qi, "Multiobjective multiverse optimizer for multirobotic u-shaped disassembly line balancing problems," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 2, pp. 882–894, 2023.

[2] X. Guo, M. Zhou, S. Liu, and L. Qi, "Multiresource-constrained selective disassembly with maximal profit and minimal energy consumption," *IEEE Transactions on Automation Science and Engineering*, vol. 18, pp. 804–816, 2021.

[3] X. Wang, M. Zhou, Q. Zhao, S. Liu, X. Guo, and L. Qi, "A branch and price algorithm for crane assignment and scheduling in slab yard," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1122–1133, 2020.

[4] C. Mingbao and S. Shijie, "Two scheduling problems in group technology with deteriorating jobs," *Applied Mathematics-A Journal of Chinese Universities*, vol. 20, pp. 225–234, 2005.

[5] M. D. Toksarı and E. Güner, "Minimizing the earliness/tardiness costs on parallel machine with learning effects and deteriorating jobs: a mixed nonlinear integer programming approach," *The International Journal of Advanced Manufacturing Technology*, vol. 38, pp. 801–808, 2008.

[6] Y. Fu, M. Zhou, X. Guo, and L. Qi, "Artificial-molecule-based chemical reaction optimization for flow shop scheduling problem with deteriorating and learning effects," *IEEE Access*, vol. 7, pp. 53 429–53 440, 2019.

[7] D.-L. Yang and W.-H. Kuo, "Some scheduling problems with deteriorating jobs and learning effects," *Computers & Industrial Engineering*, vol. 58, pp. 25–28, 2010.

[8] C. Miao, F. Kong, J. Zou, R. Ma, and Y. Huo, "Parallel-machine scheduling with step-deteriorating jobs to minimize the total (weighted) completion time," *Asia-Pacific Journal of Operational Research*, vol. 40, 2023.

[9] J.-B. Wang, D.-Y. Lv, S.-Y. Wang, and C. Jiang, "Resource allocation scheduling with deteriorating jobs and position-dependent workloads," *Journal of Industrial Management Optimization*, vol. 19, pp. 1658–1669, 2023.

[10] G. Xu, Z. Zhang, Z. Li, X. Guo, L. Qi, and X. Liu, "Multi-objective discrete brainstorming optimizer to solve the stochastic multiple-product robotic disassembly line balancing problem subject to disassembly failures," *Mathematics*, vol. 11, no. 6, p. 1557, 2023.

[11] T. Xia, G. Shi, G. Si, S. Du, and L. Xi, "Energy-oriented joint optimization of machine maintenance and tool replacement in sustainable manufacturing," *Journal of Manufacturing Systems*, vol. 59, pp. 261–271, 2021.

[12] W. Xu and L. Cao, "Optimal tool replacement with product quality deterioration and random tool failure," *International Journal of Production Research*, vol. 53, pp. 1736–1745, 2015.

[13] M. S. Salehi Mir, J. Rezaeian, and H. Mohamadian, "Scheduling parallel machine problem under general effects of deterioration and learning with past-sequence-dependent setup time: heuristic and meta-heuristic approaches," *Soft Computing*, vol. 24, pp. 1335–1355, 2019.

[14] A. Gungor and S. M. Gupta, "A solution approach to the disassembly line balancing problem in the presence of task failures," *International journal of production research*, vol. 39, pp. 1427–1467, 2001.

[15] Altekin and F. Tevhide, "A piecewise linear model for stochastic disassembly line balancing," *IFAC-PapersOnLine*, vol. 49, pp. 932–937, 2016.

[16] M. A. Ilgin, H. Akçay, and C. Araz, "Disassembly line balancing using linear physical programming," *International Journal of Production Research*, vol. 55, pp. 6108–6119, 2017.

[17] S. Hezer and Y. Kara, "A network-based shortest route model for parallel disassembly line balancing problem," *International Journal of Production Research*, vol. 53, pp. 1849–1865, 2015.

[18] S. Agrawal and M. K. Tiwari, "A collaborative ant colony algorithm to stochastic mixed-model u-shaped disassembly line balancing and sequencing problem," *International Journal of Production Research*, vol. 46, pp. 1405–1429, 2008.

[19] P. Yao and S. M. Gupta, "Fish school search optimization algorithm for solving u-shaped sequence-dependent disassembly line balancing problem with multiple objectives," *International Organization of Scientific Research*, vol. 12, pp. 10–21, 2022.

[20] Y. Zhou, X. Guo, and D. Li, "A dynamic programming approach to a multi-objective disassembly line balancing problem," *Annals of Operations Research*, vol. 311, pp. 921–944, 2022.

[21] Y. Ren, C. Zhang, F. Zhao, G. Tian, W. Lin, L. Meng, and H. Li, "Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-optimal algorithm," *Journal of Cleaner Production*, vol. 174, pp. 1475–1486, 2018.

[22] J. Liu, Z. Zhou, D. T. Pham, W. Xu, C. Ji, and Q. Liu, "Collaborative optimization of robotic disassembly sequence planning and robotic disassembly line balancing problem using improved discrete bees algorithm in remanufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101829, 2020.

[23] L. Liuke, Z. Zeqiang, Z. Binsen, and C. Ning, "Optimization of multi-objective disassembly line balancing problem using immune mechanism cooperative genetic algorithm," *Information and Control*, vol. 47, pp. 671–679, 2018.

[24] L. Zhu, Z. Zhang, and Y. Wang, "A pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation," *International Journal of Production Research*, vol. 56, pp. 7354–7374, 2018.

[25] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem," *Information Sciences*, vol. 217, pp. 65–77, 2012.

[26] G. Alp and A. F. Alkaya, "Hyperheuristic based migrating birds optimization algorithm for a fairness oriented shift scheduling problem," *Mathematical Problems in Engineering*, vol. 2021, p. 6756588, 2021.

[27] Z. Zikai, T. Qiuhua, L. Zixiang, and H. Dayong, "An efficient migrating birds optimization algorithm with idle time reduction for type-i multi-manned assembly line balancing problem," *Journal of Systems Engineering and Electronics*, vol. 32, pp. 286–296, 2021.

[28] Z. Li, M. N. Janardhanan, A. S. Ashour, and N. Dey, "Mathematical models and migrating birds optimization for robotic u-shaped assembly line balancing problem," *Neural Computing and Applications*, vol. 31, pp. 9095–9111, 2019.

[29] G. Qin, X. Guo, M. Zhou, S. Liu, and L. Qi, "Multi-objective discrete migratory bird optimizer for stochastic disassembly line balancing problem," *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 420–425, 2020.

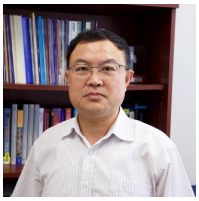
[30] M. Han, "A v2g scheduling strategy based on the fruit fly optimization algorithm," in *Journal of Physics: Conference Series*, vol. 1952, 2021, p. 042063.

[31] Y. Li, Y. He, X. Liu, X. Guo, and Z. Li, "A novel discrete whale optimization algorithm for solving knapsack problems," *Applied Intelligence*, vol. 50, pp. 3350–3366, 2020.

[32] S. Chamchuen, A. Siritaratiwat, P. Fuangfoo, P. Suthisopapan, and P. Khunkitti, "Adaptive salp swarm algorithm as optimal feature selection for power quality disturbance classification," *Applied Sciences*, vol. 11, p. 5670, 2021.

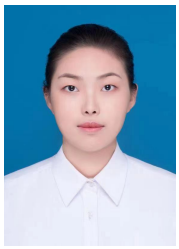
Shujin Qin (Member, IEEE) received his B.S. degree in Information and Computing Science from Tianjin Polytechnic University, Tianjin, China in 2008, M.S. degree in Operational Research and Cybernetics from University of Science and Technology Liaoning, Anshan, China in 2011, and Ph.D. degree in System Engineering from Northeastern University, Shenyang, China in 2019. He joined Shangqiu Normal University, Shangqiu, China in 2019, and is now a lecturer of Information and Technology. His current research interests include large-scaled integer programming, cutting stock problem, vehicle routing problem, traveling repairman problem and intelligent optimization algorithm.

Feng Guo received his B.S. degree in Aviation Transportation Management from Civil Aviation University of China, Tianjin, China, in 2021. He is currently a Graduate student of the Computer Science and Technology at Liaoning Petrochemical University, Fushun, China.



Jiacun Wang (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Nanjing University of Science and Technology (NUST), China, in 1991. He is currently a Professor of software engineering at Monmouth University, West Long Branch, New Jersey, USA. From 2001 to 2004, he was a member of scientific staff with Nortel Networks in Richardson, Texas. Prior to joining Nortel, he was a research associate of the School of Computer Science, Florida International University at Miami. His research interests include software engineering, discrete event systems, formal methods, machine learning, and real-time distributed systems.

Liang Qi (Member, IEEE) received his B.S. degree in Information and Computing Science and M.S. degree in Computer Software and Theory from Shandong University of Science and Technology, Qingdao, China, in 2009 and 2012, respectively, and Ph.D. degree in Computer Software and Theory from Tongji University, Shanghai, China in 2017. He is currently a lecturer of Computer Science and Technology at Shandong University of Science and Technology, Qingdao, China. From 2015 to 2017, he was a visiting student in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. His current research interests include Petri nets, discrete event systems and optimization algorithms.



Jiaxin Wang received her B.S. degree in Computer Science and Technology from Henan University of Chinese Medicine, Zhengzhou, China in 2021. She is currently a Graduate student of the College of Computer and Communication Engineering at Liaoning Petrochemical University, Fushun, China. Her research interests are in intelligent optimization algorithm.

Xiwang Guo (Senior Member, IEEE) received his B.S. degree in Computer Science and Technology from Shenyang Institute of Engineering, Shenyang, China, in 2006, M.S. degree in Aeronautics and Astronautics Manufacturing Engineering. from Shenyang Aerospace University, Shenyang, China, in 2009, Ph.D. degree in System Engineering from Northeastern University, Shenyang, China, in 2015. He is currently an associate professor of the College of Computer and Communication Engineering at Liaoning Petrochemical University. From 2016 to 2018, he was a visiting scholar of Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA.

Ziyan Zhao (Member, IEEE) received his B.S. and M.S. degrees in 2015 and 2017, respectively, from the Department of Information Science and Engineering, Northeastern University, Shenyang, China, where he is currently working toward his Ph.D. degree. From October 2018 to October 2020, he worked as a visiting Ph. D. student in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA., supported by a scholarship from the China Scholarship Council. His research focuses on production planning and scheduling, intelligent manufacturing, industrial big data, and intelligent optimization algorithm. Till now, he has published over 10 international journal and conference papers in the above areas.