Mathematical formulation and an improved moth-flame optimization algorithm for parallel two-sided disassembly line balancing based on fixed common stations

Yu Zhang[a,b], Zeqiang Zhang[a,b,*], Tao Yin[a,b], Wei Liang[a,b]

[a] *School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China*

[b] *Technology and Equipment of Rail Transit Operation and Maintenance Key Laboratory of Sichuan Province, Chengdu 610031, China*

Abstract: Nowadays, rapid product iterations result in large quantities of end-of-life products. To meet the fast-growing demand for remanufacturing engineering, companies have quickened the standardization and industrialization of waste dissembling. Two-sided disassembly lines can effectively disassemble large-sized products on both sides of the lines, and parallel disassembly lines can disassemble multiple products simultaneously with fewer workstations and higher production efficiency. Combining the two types of disassembly can effectively increase the disassembly efficiency of large-sized products. However, the parallel two-sided disassembly line has not been fully investigated because of the essential complexity of the problem. Therefore, this research introduced the parallel two-sided disassembly lines balancing problem (PTDLBP) with fixed common stations. Firstly, a multi-objective mixed-integer programming model is established to solve the problem for the first time. The model is proved to be correct through small-scale numerical examples. Secondly, a multi-objective improved moth–flame optimization algorithm is implemented to solve the proposed large-scale problems. The proposed algorithm employs a two-phase decoding approach to design the scheme and a discrete moth for fire operation to search and replace new individuals, then a restart strategy is introduced to reduce the probability of the population falling into a local optimum. Finally, the algorithm solved extensive disassembly line balancing problems with different layouts including the straight-line, two-sided, and parallel two-sided, and case studies demonstrated the reliability and validity of the proposed method.

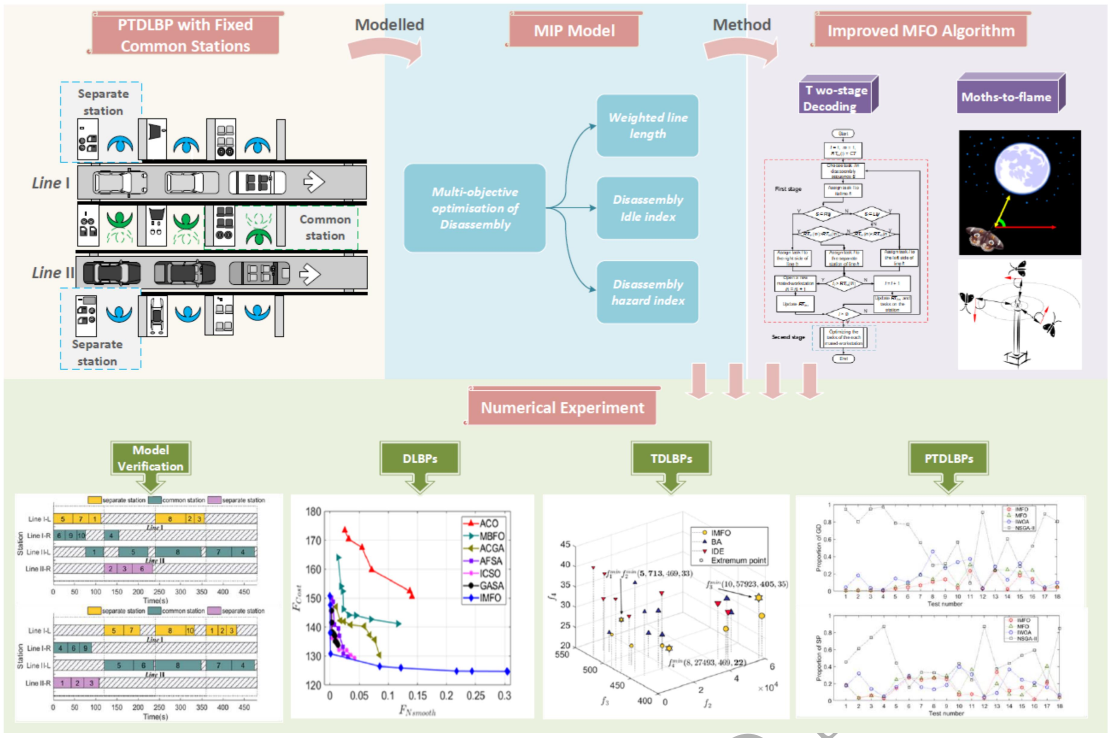
ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

© The Author(s) 2022. Published by Oxford University Press on behalf of the Society for Computational Design and Engineering. This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (https://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

Graphical abstract

Mathematical formulation and an improved moth-flame optimization algorithm for parallel two-sided disassembly line balancing based on fixed common stations



Keywords: Disassembly line balancing; Parallel two-sided disassembly line; Mixed-integer programming model; Moth-flame optimization algorithm; Multi-objective optimization

Highlights:

- The parallel two-sided disassembly line balancing problem (PTDLBP) is considered.
- An MIP model for PTDLBP is proposed for the first time.
- An improved moth-flame optimization algorithm for DLBPs is proposed.
- The validity of the proposed method is demonstrated by numerous cases.

1. Introduction

The recycling and disassembly of waste electronic products is a crucial part of the development of a global circular economy. This not only effectively avoids the damage caused by long-term storage of waste products to the surrounding atmosphere, water, soil, and ecological systems, but also alleviates the bottlenecks in development caused by resource shortages in countries. The disassembly line is the first step and the most effective means to recycle many products. Balancing disassembly lines has an important impact on the overall performance of an entire remanufacturing system. The problem of optimizing the disassembly task allocation in a manner that multiple goals are optimized effectively, including the number of workstations, load of each station, harmful degree of the production line, etc., is called the disassembly line balancing problem (DLBP) (Gungor & Gupta, 2001).

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

Depending on the different purposes there exist several configurations in DLBPs. Large-sized waste products such as automobiles are often disassembled in the form of two-sided disassembly lines. Two-sided disassembly lines have operators on both sides of a conveyor belt, and two opposite stations operate in parallel simultaneously on the same product. Two-sided disassembly lines are productive for large-sized waste products, effectively achieving shorter line lengths, reduced production times, fewer worker movements, and higher production efficiency (Zhang et al., 2022).

For a large number of end-of-life products, the traditional single disassembly line may not fulfil the disassembly efficiency requirements. So Hezer & Kara (2015) proposed the concept of the parallel disassembly line balancing problem (PDLBP), which uses two or more disassembly lines parallel to each other. Compared with single-line disassembly lines, parallel disassembly lines can disassemble similar products or different models of the same products at different cycle times in adjacent lines (Bhosale & Pawar, 2020). Workers between the two lines can manage tasks on both lines. This layout effectively results in fewer open workstations, improved resource utilization, and higher production productivity (Zhu et al., 2020). Aiming at handling a mass of large waste products, the parallel two-sided disassembly line, which mixes the advantages of the two layouts mentioned above, has become one of the choices for decision makers.

The parallel two-sided disassembly line is a manufacturing mode in which two or more two-sided disassembly lines are parallel to each other, aiming to produce one or more products with similar disassembly tasks, as shown in Fig. 1. Stations on parallel two-sided disassembly lines can be divided into separate and common stations. The stations adjacent to the two lines can be merged into common stations. Operators at common stations must work on both the right and left sides. If common stations are fixedly placed on two neighboring lines, the generality of the overall structure can be improved. In this condition, for many large-sized products, only a few operators would be required, and disassembly efficiency can be increased. However, when balancing two-sided disassembly lines, owing to the complex constraints of priority relationships, idle time is occasionally unavoidable, and when it comes to two or more two-sided disassembly lines, the complexity of problems increases significantly. Beyond that, DLBPs are NP-hard combinatorial optimization problems (McGovern & Gupta, 2007), and multi-objective collaborative optimization should also be considered. Under those circumstances, as the necessity and difficulty of the PTDLBP, it is certainly worth developing powerful and efficient ways to reach the optimal solution that meets the demand.

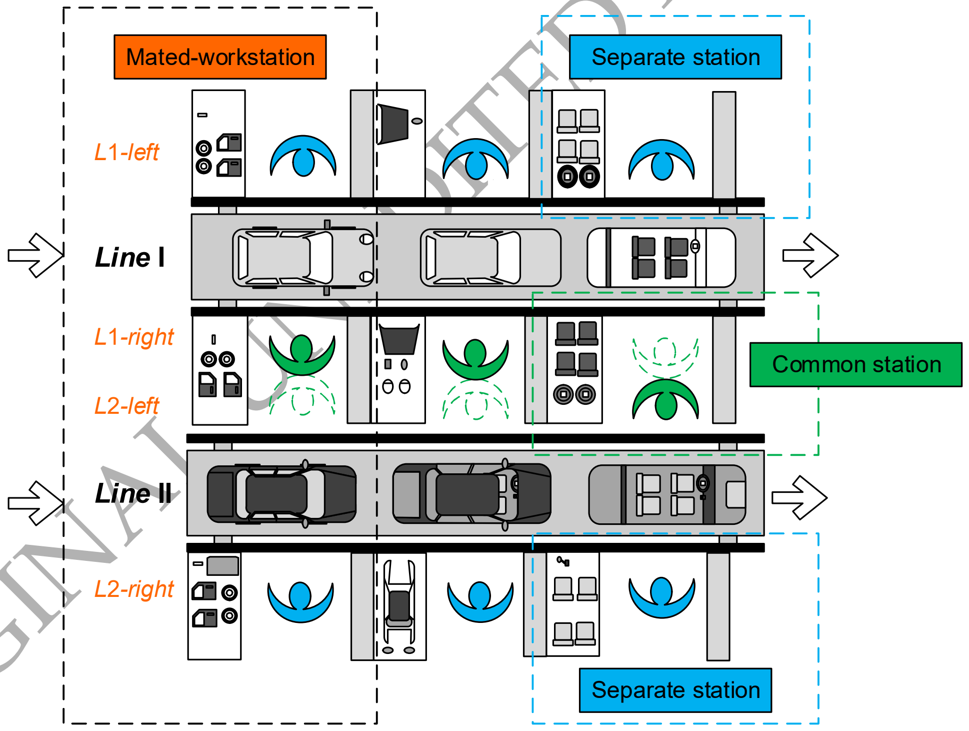


Fig. 1 Configuration of a parallel two-sided assembly line

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

2. Literature review

The DLBP has attracted significant interest from scholars since it was first proposed in 2001. Gungor et al. (2001) firstly established a complete disassembly model with five optimization objectives, including minimized idle time and disassembly direction changes and prioritized disassembly of easily accessible, hazardous, and highly demanded parts, which lay a theoretical foundation for later study. We review the evolvement of DLBP from the layout types and solving methods and point out the insufficiency in the following part.

2.1 Line style

The disassembly lines have diverse layouts due to different characteristics. Research on DLBPs mainly concerns the structure of the straight, U-shaped, two-sided, and parallel (Özceylan et al., 2019). Among them, the most basic straight type, to which most attention has been paid, can promote standardization and larger-scale operations (Edis et al., 2022; Wu et al., 2022). The U-shaped line uses less area and is more efficient than the straight lines (Li & Janardhanan, 2021; Wang et al., 2020).

Two-sided disassembly lines apply to the disposal of large-scale end-of-life products and have attracted increasing interest. Wang et al. (2019) developed a flower pollination algorithm for a stochastic two-sided partial DLBP. Kucukkoc (2020) established a mixed-integer linear programming model for the TDLBP for the first time. Considering the energy consumption objective, Liang, et al. (2021) solved the two-sided disassembly line model with a parallel operation. Zhang et al. (2022) considered the part characteristic indexes, established a multi-objective MIP for two-sided disassembly lines and proposed an improved whale optimization algorithm to solve the considered problem. Mutlu & Güner (2021) reported a memetic algorithm to solve the balancing problem of mixed-model two-sided disassembly lines. All the above studies were conducted for a single two-sided disassembly line.

To increase the productivity of a line and reduce disassembly costs, Hezer & Kara (2015) first defined and solved a PDLBP using a network model based on the shortest route model. Zhu, Zhang, & Guan, (2020) studied the parallel partial disassembly line balancing problem (PPDLBP) with hazardous and demand parts. Wang et al. (2021) studied the PPDLBP considering the uncertainty of task time and variables and then developed a genetic simulated annealing algorithm to solve it. These studies were all conducted for multiple straight linear disassembly lines.

2.2 Methodologies

There are three main solution approaches in DLBP studies, including exact methodologies, heuristic, and meta-heuristic approaches. The early common methods to solve DLBPs were exact algorithms, including integer programming (Altekin et al., 2008), dynamic programming (Koc et al., 2009), and piecewise linear programming (Altekin, 2017). Because the DLBP is an NP-complete problem (McGovern & Gupta, 2007), feasible solutions increase geometrically with an increase in problem scales, and the solution effect of the exact methods are greatly reduced. So heuristic and meta-heuristic algorithms are also widely used. For heuristic algorithms, such as uninformed (H-K) search methods (McGovern & Gupta, 2005) and reinforcement learning (Tuncel et al., 2014), because they are specially designed for the particular problem, so their generality is not good.

The meta-heuristic strategy is frequently a general strategy that does not depend on the unique conditions of a problem and has good generality and strong global optimization ability (Shu et al., 2022). Refer to, for example, Kalayci & Gupta (2013) for the artificial bee colony, Kalayci & Gupta (2014) for the taboo search algorithm, and Kalayci et al. (2015) for the variable neighborhood search. The above algorithms sort the optimization objective through dictionary sorting, and multi-objectives are managed in turn according to the assumed importance, which transforms the multi-objective problem into a single objective one, and a single optimal solution can be obtained. Ding et al. (2009) introduced Pareto optimality into the ant colony algorithm and provided multiple feasible solutions when a decision-maker's preference is unknown.

2.3 Knowledge Gaps

Interestingly, after looking back at studies of DLBPs, no research was conducted on the parallel two-sided disassembly line balancing problem, although there are few reports on the parallel two-sided assembly line balancing problems (PTALBPs) (Kucukkoc & Zhang, 2014; Özcan et al., 2010). However, DLBPs are not a simple inversion of ALBPs (Lambert & Gupta, 2008), and there are many differences between ALBPs and DLBPs (e.g. precedence relationship and optimization objectives

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

(Koc et al., 2009). Therefore, one of the motivations of this study was to make a supplement in this field. It should also be mentioned that literature about PTALBP rarely provided solvable mixed integer programming (MIP) models and supposed that common stations are not fixed. The current study considers comprehensively above-mentioned factors, and the main contributions are as follows:

- This paper contributes to the references by introducing the parallel two-sided disassembly line balancing problem with fixed common stations.
- A MIP model for PTDLBP is proposed considering the opening of common stations between two adjacent lines.
- An improved moth-flame optimization algorithm (IMFO) is proposed, and special discretization operations are presented.
- The validity of the proposed algorithm is demonstrated by comparison with MIP and other algorithms.

The remainder of this paper is organized as follows. Section 3 introduces more details of the PTDLBP and presents the mathematical formulation. Section 4 describes the proposed IMFO process. Section 5 provides verification and demonstration, including model verification and the performance of the algorithm on DLBPs, TDLBPs, and PTDLBPs. Section 6 summarizes the research and presents the future research considerations.

3. Problem description and formulation

3.1 Problem description

Parallel two-sided disassembly lines have multiple two-sided disassembly lines in parallel, which can disassemble one or more product models with similar characteristics. Skilled workers are assigned on both sides of each conveyor belt to perform the corresponding disassembly tasks at their respective stations. Each line has a left and right station. Each pair of workstations of all lines together form a mated-workstation. Unlike two separate two-sided disassembly lines, two stations between two adjacent lines are fixedly combined into a common station (see Fig. 1). The right station of Line I and the left station of Line II are combined into a single station called the common station. For common stations, only one worker is required to complete the tasks assigned to the station, and the workers must complete the tasks of both lines by moving back and forth. The left station of Line I and the right station of Line II are called separate stations. For separate stations, the workers must fulfil the tasks on one side of any parallel two-sided line.

The types of disassembly tasks include R-type tasks (can only be assigned to the right), L-type tasks (can only be assigned to the left), and E-type tasks (can be assigned to both sides). Each disassembly task has an operating time, and the total task time in each station cannot exceed an upper time limit called the cycle time (CT). In addition, the priority relationship between tasks should be carefully considered, which results in extra idle time. Fig. 2 shows an example of a parallel two-sided disassembly line. In Line I, task 4 immediately precedes task 6, and task 2 can only be assigned to the left, which results in a gap between tasks 2 and 4. The shaded part between tasks 4 and 2 in Fig. 2 is the idle time caused by interference.

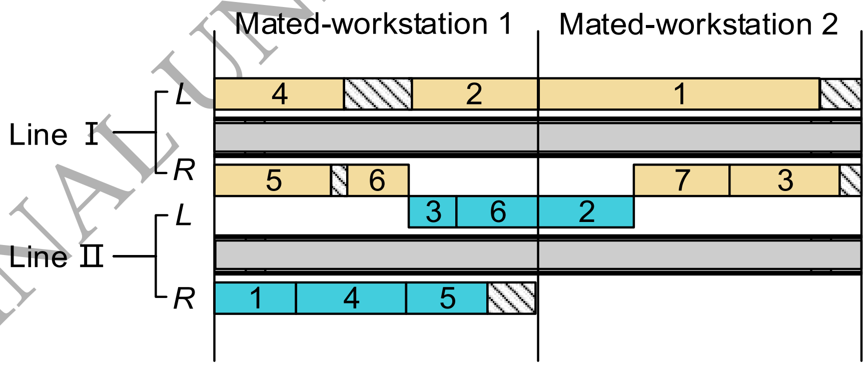


Fig. 2 Task allocation of a parallel two-sided disassembly line

In addition, different product models may have different cycle times, and it is difficult to satisfy the time constraints for a common station. Therefore, a method of the least common multiple (LCM) was introduced (Gökçen et al., 2006). This method

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

adopts the common CT as a uniform cycle. The specific methods are as follows: (a) Determine the LCM of the line CTs. (b) Obtain D_1 and D_2 by dividing both CTs by the LCM value. (c) Multiply D_1 and D_2 by the processing time of each task on lines 1 and 2, respectively. (d) Select the LCM as the common CT of the lines.

Considering the abovementioned constraints, an MIP model is developed for parallel two-sided disassembly lines. Recycling factories focus on the benefits, efficiency, and harmfulness of disassembly. Consequently, the optimization objectives of PTDLBP in this paper are to minimize the weighted line length, load balance, and hazard index. The following statements are assumed in this paper.

- Each line can disassemble different products.
- Each disassembly line may have a different CT.
- The precedence diagrams for each product model are known.
- The tasks performed are deterministic and operate within a known time and harmfulness coefficient.
- Workers moving times in stations are ignored.
- Parallel tasks and parallel stations cannot be permitted.

3.2 Mathematical model

The notations used in the mathematical model proposed in this paper can be summarized as follows:

i, j, s:	Disassembly task indexes
$\boldsymbol{L}_h$:	The h^{th} line. $h = \{1, 2, \dots, H\}$
k:	Parallel mated-workstations index; 2 separate stations and h - 1 common station form parallel mated-workstations
n:	Number of disassembly tasks for all lines
m:	Number of available parallel mated-workstations
$\boldsymbol{I}$:	Disassembly task index set for all lines
$\boldsymbol{M}$:	Parallel mated-workstations index set
$\boldsymbol{P}$:	Station of the mated-workstations, $\boldsymbol{P} = \{p \mid 1, 2, \dots, H + 1\}$; $p = 1$ for the left separate station of line 1, $p = H + 1$ for the right separate station of line h, $p = h$ ($h=\{2, 3, \dots H\}$) for the common station of between line h-1 and line h
$\boldsymbol{Rty}$:	Set of R-type tasks
$\boldsymbol{Lty}$:	Set of L-type tasks
t_i:	Processing time of disassembly task i
d_i:	Demand index of disassembly task i; if the task has a demand attribute, d_i is a natural number, otherwise $d_i = 0$
h_i:	Hazard index of disassembly task i; if the task has a hazard attribute, $h_i = 1$, otherwise $h_i = 0$
CT:	Common cycle time of the disassembly line
$\boldsymbol{A}_{ij}$:	Priority matrix for disassembly tasks, $\boldsymbol{A}_{ij} = [a_{ij}]_{n\times n}$; if a_{ij}=1, task i is an immediately preceding task for task j
ξ:	A large real number, $\xi = CT \cdot n$
w_i:	Start time of task i, $w_i \geqslant 0$
x_{ik}^p:	Task assignment variable; if task i is assigned to the p-side of mated-workstations k, $x_{ik}^p = 1$, otherwise, $x_{ik}^p = 0$
z_{ijk}^p:	Disassembly sequence variable; if tasks i and j are assigned to the p-side of mated-workstations k and task i is assigned before task j, $z_{ijk}^p = 1$, otherwise, $z_{ijk}^p = 0$
S_k:	Parallel mated-workstations variable; if mated-workstations k is open, $S_k = 1$, otherwise, $S_k = 0$
U_k^p:	Station variables; if station p of mated-workstations k is open, $U_k^p = 1$, otherwise, $U_k^p = 0$

Objective:

$$MinF_1 = \gamma_1 \cdot \sum_{k=1}^{m} S_k + \gamma_2 \cdot \sum_{k=1}^{m} \sum_{p\in \boldsymbol{P}} U_k^p \tag{1}$$

$$MinF_2 = \sum_{k=1}^{m} \left(H \cdot S_k \cdot CT - \sum_{i=1}^{n} \sum_{p\in \boldsymbol{P}} x_{ik}^p \cdot t_i \right)^2 \tag{2}$$

$$MinF_3 = \sum_{i=1}^{n} h_i \cdot w_i \tag{3}$$

Subject to:

$$\sum_{k=1}^{m} \sum_{p\in \boldsymbol{P}} x_{ik}^p = 1, \forall i \in \boldsymbol{I} \tag{4}$$

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

$$\sum_{k=1}^{m}\sum_{p=h}^{h+1} x_{ik}^{p} = 1, \forall i \in \boldsymbol{L}_h \tag{5}$$

$$\sum_{k=1}^{m} x_{ik}^{h} = 1, \forall i \in \boldsymbol{L}_h \wedge i \in \boldsymbol{Lty} \tag{6}$$

$$\sum_{k=1}^{m} x_{ik}^{h+1} = 1, \forall i \in \boldsymbol{L}_h \wedge i \in \boldsymbol{Rty} \tag{7}$$

$$\left\lceil \sum_{i=1}^{n} t_i \Big/ (H \cdot CT) \right\rceil \le \sum_{k=1}^{m} S_k \le n, \forall i \in \boldsymbol{I} \tag{8}$$

$$S_k \le S_{k-1}, \forall k \in \{2,...,m\} \tag{9}$$

$$CT \cdot \left[\left(\sum_{k=1}^{m} \sum_{p\in\boldsymbol{P}} x_{ik}^{p} \cdot k \right) - 1 \right] + \sum_{k=1}^{m} \sum_{p\in\boldsymbol{P}} \sum_{s\in\boldsymbol{I}, s\neq i} z_{sik}^{p} \cdot t_s \le w_i, \forall i \in \boldsymbol{I} \tag{10}$$

$$w_i + t_i \le \left(\sum_{k=1}^{m} \sum_{p\in\boldsymbol{P}} x_{ik}^{p} \cdot k \right) \cdot CT, \forall i \in \boldsymbol{I} \tag{11}$$

$$x_{ik}^{p} + x_{jk}^{p} \le 1 + (z_{ijk}^{p} + z_{jik}^{p}), 1 \le i < j \le n; \forall p \in \boldsymbol{P}; k \in \boldsymbol{M} \tag{12}$$

$$\frac{1}{2} \cdot (x_{ik}^{p} + x_{jk}^{p}) \ge z_{ijk}^{p} + z_{jik}^{p}, 1 \le i < j \le n; \forall p \in \boldsymbol{P}; k \in \boldsymbol{M} \tag{13}$$

$$\xi \cdot (1 - z_{jik}^{p}) + w_i \ge w_j + t_j, \forall i, j \in \boldsymbol{I}; i \neq j; p \in \boldsymbol{P}; k \in \boldsymbol{M} \tag{14}$$

$$w_i \ge w_j + t_j, \forall i, j \in \boldsymbol{I} \wedge \{i, j \mid \boldsymbol{A}_{ji} = 1\} \tag{15}$$

$$S_k \le \sum_{p\in\boldsymbol{P}} \sum_{i=1}^{n} x_{ik}^{p} \le n \cdot S_k, \forall k \in \boldsymbol{M} \tag{16}$$

$$S_k \le \sum_{p\in\boldsymbol{P}} U_k^{p} \le (H+1) \cdot S_k, \forall k \in \boldsymbol{M} \tag{17}$$

$$U_k^{p} \le \sum_{i=1}^{n} x_{ik}^{p} \le n \cdot U_k^{p}, \forall k \in \boldsymbol{M}; p \in \boldsymbol{P} \tag{18}$$

$$x_{ik}^{p} = \{0,1\}, \forall i \in \boldsymbol{I}; p \in \boldsymbol{P}; k \in \boldsymbol{M} \tag{19}$$

$$z_{ijk}^{p} = \{0,1\}, 1 \le i < j \le n; \forall p \in \boldsymbol{P}; k \in \boldsymbol{M} \tag{20}$$

$$S_k = \{0,1\}, \forall k \in \boldsymbol{M} \tag{21}$$

$$U_k^{p} = \{0,1\}, \forall k \in \boldsymbol{M}; p \in \boldsymbol{P} \tag{22}$$

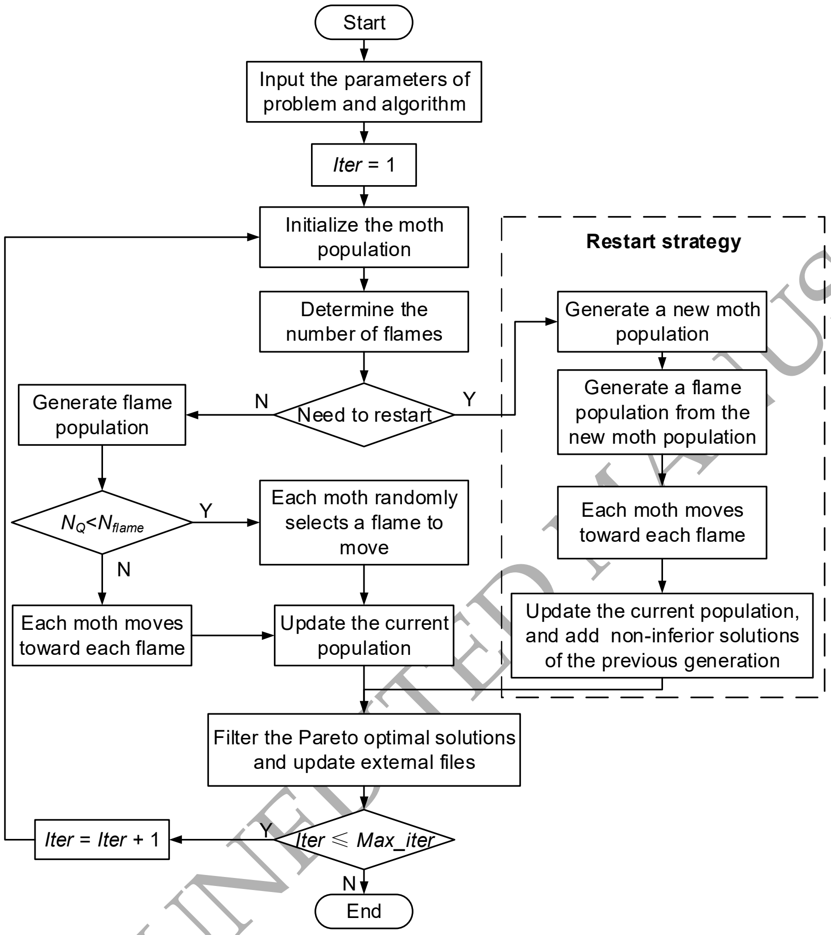
The three primary objectives that we focus on are formulated in Eqs. (1)–(3). Eq. (1) indicates the minimum number of weighted sums of mated-workstations (line length) and stations. With reference to Kucukkoc (2020), γ_1 and γ_2 are set to 100 and 1, respectively. Eq. (2) represents the idle index to balance the workload at each mated-workstation. This objective function is calculated using the total task time in the mated-workstations rather than in the stations, which reduces the line length while balancing the idle time (Zhang et al., 2022). Eq. (3) attempts to minimize the harmfulness of the disassembly; hazardous parts should be disassembled as early as possible to reduce the impact on the environment and workers' health. Eq. (4) constraints each task to be assigned to a certain station. Eq. (5) restricts tasks to being assigned to specific lines. Eqs. (6) and (7) address the direction constraints, R-type tasks to the right, and L-type tasks to the left. Eqs. (8) and (9) describe the mated-workstation, including limiting the number range of mated-workstations and opening mated-workstations in turn. Eqs. (10) and (11) address the processing time of each task and define the start and end times of tasks. Eqs. (12) and (13) establish linkages between x_{ik}^{p} and z_{ijk}^{p}. Eq. (14) must be satisfied between the front and back tasks assigned to the same station. Eq. (15) ensures that the priority constraint is satisfied. Eqs. (16)–(18) set up a corresponding relationship between x_{ik}^{p}, U_k^{p}, and S_k. Each open mated-workstation and station should be assigned to at least one task, and the number of tasks allocated should not exceed the total number of tasks. Eqs. (19)–(22) introduce various decision variables.

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

4. Improved moth-flame optimization algorithm

This section introduces the improved MFO-based approach, called the IMFO, which is proposed to solve the PTDLBP. The MFO algorithm (Mirjalili, 2015) is a swarm intelligence optimization method, and its working flow simulates the behavior of moths plunging into a flame. The algorithm has a simple structure and fewer parameters and is designed to solve continuous optimization problems. However, it has also been successfully adopted to solve combinatorial optimization, such as permutation-based problems (Helmi & Alenany, 2020). Considering its effective application in other discrete optimization problems and no application in the DLBP area, we attempt to design the MFO algorithm for solving DLBPs for the first time. The flow chart of the IMFO algorithm is shown in Fig. 3, where the dotted box indicates the restart strategy flow. The algorithmic details are given in the sections that follow.



Start
Input the parameters of problem and algorithm
Iter = 1
Initialize the moth population
Determine the number of flames
Need to restart
N
Y
Generate flame population
N_Q<N_{flame}
Y
N
Each moth randomly selects a flame to move
Each moth moves toward each flame
Update the current population
Restart strategy
Generate a new moth population
Generate a flame population from the new moth population
Each moth moves toward each flame
Update the current population, and add non-inferior solutions of the previous generation
Filter the Pareto optimal solutions and update external files
Iter = Iter + 1
Y
Iter ≤ Max_iter
N
End

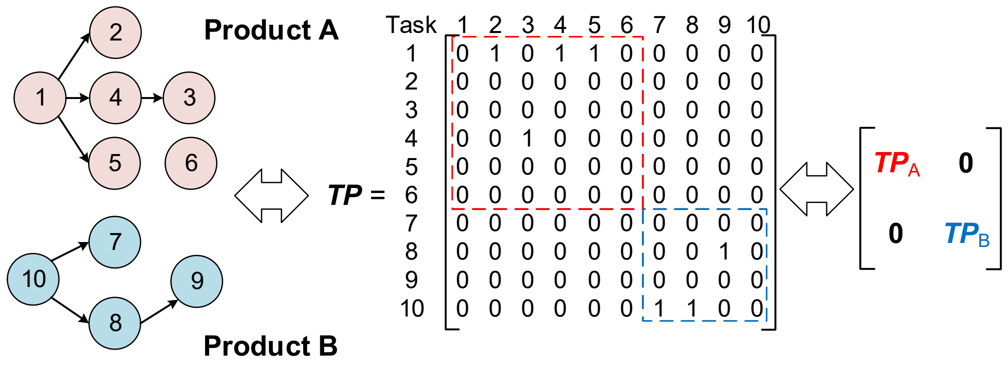
Fig. 3 Flow chart of the IMFO algorithm

4.1. Encoding

Because of the discreteness of the PTDLBP, this paper adopts real coding. The initial solutions are generated randomly under priority constraint conditions to ensure the diversity of the population. First, multiple product priority relationship matrices are obtained through merging. Fig. 4 shows a combined priority matrix for the two unprocessed products. $\boldsymbol{TP}_{[i,j]} = 1$ indicates that task i is the immediate task of task j. The red dotted box indicates the priority relation matrix for a product with six tasks, and the blue dotted box indicates the priority relation matrix for a product with four tasks. The priority matrix of more products can be combined according to this rule. Individual initialization is implemented using Algorithm 1.

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022



$$TP = \begin{array}{c|cccccccccc}
\text{Task} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{array} \Leftrightarrow \begin{bmatrix} TP_A & 0 \\ 0 & TP_B \end{bmatrix}$$

Fig. 4 Combined priority matrices for parallel two-sided disassembly lines

```
Algorithm 1: Initialize individual generation
Input:    number of all tasks (T_SIZE), priority relation array (TP)
Output:   initial population (individual)
1   $i = 1$;
2   $individual$ = zeros(1, $T\_SIZE$);
3   While $i \le T\_SIZE$
4       Randomly select a zero vector $s$ in column vectors of TP column vector $s$ with a zero
        vector in TP and set $individual(i) = s$; // Randomly select a task $s$ with no predecessors
        and set task $s$ as the $i^{th}$ position of the individual;
5       Set all elements in row $s$ in TP to 0, and elements in column $s$ in TP to 1; // Relax the
        priority constraint of task $s$ and prevent task $s$ from being selected again;
6       $i = i + 1$;
7   End While
```

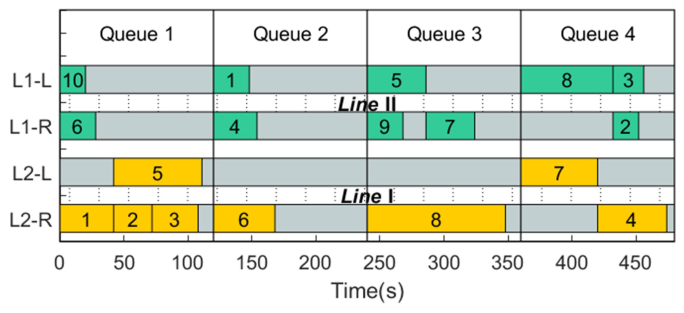
4.2. Decoding

There is no information on the direction of task assignment in the coding method, which must be considered during decoding. A two-stage heuristic decoding method is adopted in this paper. The aim of the first-stage decoding is to assign tasks to each mated-workstation. At this stage, the tasks are assigned according to the disassembly priority sequence. First, the line to which a task belongs is determined; therefore, the task can only be assigned to the corresponding line. Subsequently, the disassembly direction of the task is determined, and the allocation position is further limited. Finally, the time available for this task is calculated in the latest open mated-workstation. $Available\_time = \mathrm{CT} \cdot k - \max(Elapsed\_time1, Elapsed\_time2)$, where k represents the current mated-workstation index and *Elapsed_Time 1* indicates the end time of the latest task in the current station. *Elapsed_Time 2* represents the end time of the last immediately preceding task assigned to the other side of the line. Note that for E-type tasks, priority is assigned to the side with more available time. If both sides have the same available time, they are assigned to separate stations. If the available time is insufficient for disassembly, then open a new mated-workstation.

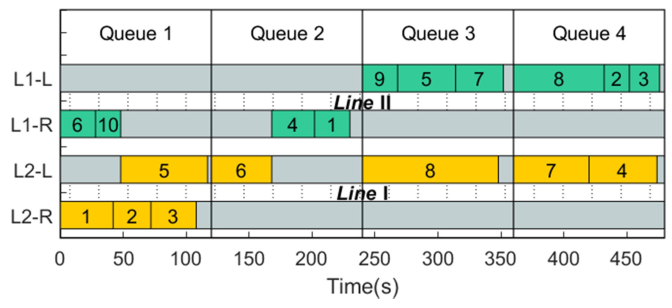
Many deficiencies may be observed based on the first stage of the decoding method, such as the unbalanced workload of tasks or redundancy of stations. Therefore, we optimize the solution of the first stage in the second stage. Tasks in each mated-workstation are reassembled and assigned separately to unbalanced assignment problems, which effectively reduces the number of open stations for the first-stage allocation scheme. The heuristic method for the two parallel lines is illustrated in Algorithm 2 as follows.

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022



(a) Unoptimized decoding scheme



(b) Optimized decoding scheme

Fig. 5 Decoding optimization

Algorithm 2: Mated-workstation optimization

Input: ***task_Assigned_Matrix*** (Decoding scheme of the mated-workstation obtained in the first stage), ***task_Data*** (Priority relation matrix and disassembly task information), ***S*** (Disassembly sequence)

Output: ***task_Assigned_Matrix*** (Optimized decoding scheme of one mated-workstation)

// Each line is divided into left and right sides and two lines have a total of four sides.

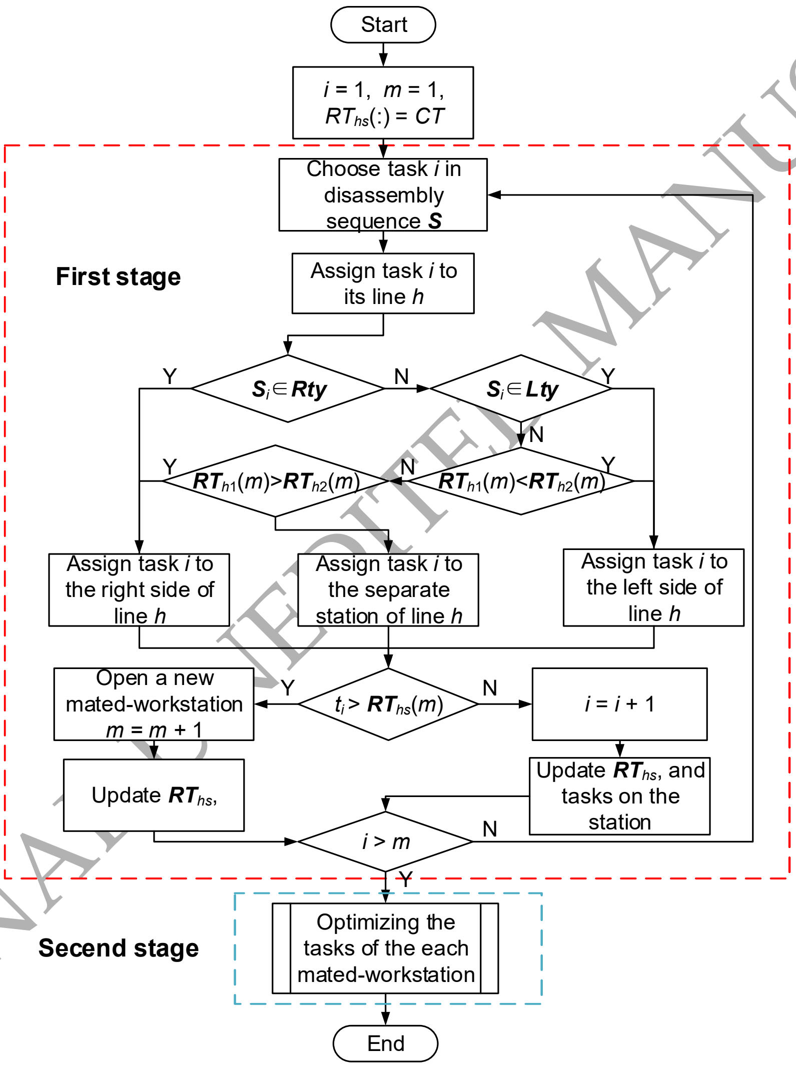
```
//————————————————————————————————————————————
1  If tasks on each line can be assigned to the common station and satisfy time and direction constraints
2  | Do assign all tasks to the common station and update allocation results
   | Return
3  End If
//————————————————————————————————————————————
4  If three sides have tasks
5  | If tasks in the line with tasks on both sides can be assigned to the same side and satisfy time and direction constraints
6  | | Do assign the tasks in the line with tasks on both sides to the same side and update task_Assigned_Matrix
7  | | Return
8  | End If
9  | If tasks of the line with tasks on one side can be assigned to the same side and satisfy direction constraints
10 | | For $i = 0 : Ne$ // $Ne$ refers to the number of E-type tasks of the common station for line with tasks on both sides
10 | | | Do assign all tasks of the line with tasks on one side to the common station and assign the first $i$ E-type tasks of the common
   | | |      station for line with tasks on both sides to the other side. Subsequently, assign them in sequence according to the order of S
11 | | | If assigned solutions satisfy time constraints
12 | | | | Do update task_Assigned_Matrix
13 | | | | Return
14 | | | End If
15 | | End For
16 | End If
17 End If
//————————————————————————————————————————————
18 If four sides have tasks
19 | If tasks in each line can be assigned to a separate station and satisfy time and direction constraints
20 | | Do update task_Assigned_Matrix
21 | | Return
22 | End If
23 | For $h = 1 : 2$
24 | | If tasks in line $h$ can be assigned to the common station
```

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

```
25         For $i = 0 : Ne$ // $Ne$ refers to the number of E-type tasks of the common station for line opposite to line $h$
26             Do assign all tasks of the line $h$ to the common station and the first $i$ E-type tasks of the common station for line opposite to
                   line $h$ to the other side in sequence according to the order of $\boldsymbol{S}$
27             If assigned solutions satisfy time constraints
28                 Do update task_Assigned_Matrix
29                 Return
30             End If
31         End For
32     End If
33   End For
34 End If
```

Fig. 5 depicts the visualization effect of the method. Fig. 5a shows a disassembly scheme obtained using the first stage decoding, and Fig. 5b shows the optimized scheme. Twelve stations are opened in the non-optimized decoding scheme, and only seven stations are opened in the optimized decoding scheme. This significantly improves the decoding effect. The entire decoding scheme flow is shown in Fig. 6, where $\boldsymbol{S}$ is the disassembly sequence, i is the index of the disassembly sequence, m is the number of mated-workstations, and $\boldsymbol{RT}_{hp}$ is the remaining available time in the side p of line h, where $p = 1$ refers to the right side and $p = 2$ refers to the left side.



Start
i = 1, m = 1, RT_hs(:) = CT
Choose task i in disassembly sequence S
First stage
Assign task i to its line h
S_i∈Rty
S_i∈Lty
Y
N
RT_h1(m)>RT_h2(m)
RT_h1(m)<RT_h2(m)
Assign task i to the right side of line h
Assign task i to the separate station of line h
Assign task i to the left side of line h
Open a new mated-workstation m = m + 1
t_i > RT_hs(m)
i = i + 1
Update RT_hs,
Update RT_hs, and tasks on the station
i > m
Secend stage
Optimizing the tasks of the each mated-workstation
End

Fig. 6 Decoding process

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

4.3 Initialization of population

In the proposed MFO algorithm, moths represent the main body for spiral flight, while flames represent the best position the moths have found thus far. The moth population is set as an $m \times n$ matrix, and the array *OM* is adopted to store the fitness value corresponding to *M*. Similarly, the flame population *F* is initialized as a matrix of m × n, and the array *OF* is adopted to store the fitness value corresponding to *F*, expressions as shown in Eqs. (23) and (24), where *m* indicates the number of moths and *n* indicates problem scales. The initial *M* and *OM* are obtained by initializing (Section 4.1) and decoding (Section 4.2). The Pareto solution set of the *OM* is stored in *F*. If the size of *F* is less than *m*, positions are randomly generated to fill in *F* and update *OF*.

$$M=\begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ M_{21} & M_{22} & \cdots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mn} \end{bmatrix}, F=\begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1n} \\ F_{21} & F_{22} & \cdots & F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ F_{m1} & F_{m2} & \cdots & F_{mn} \end{bmatrix} \tag{23}$$

$$OM=\begin{bmatrix} OM_1 & OM_2 & \cdots & OM_m \end{bmatrix}^T, OF=\begin{bmatrix} OF_1 & OF_2 & \cdots & OF_m \end{bmatrix}^T \tag{24}$$

4.4 Location update mechanism

This section introduces the location update mechanism of the proposed MFO algorithm, including moths-to-flame and flame renewal. Moths are phototactic. While searching for a light at night, they determine their next flight by calculating their relative position to the light (frequently the moonlight). When encountering a close light such as a flame, the moth cannot fly along a straight line but moves along a spiral track to reach the flame. During the optimization process, if a better location is determined, the corresponding flame is updated, which prevents MFO from losing the current optimal solution during operation.

I. Moths-to-flame

Moths fly toward the flame following a logarithmic spiral function, and the mathematical description is shown in Eq. (25).

$$M_i = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j \tag{25}$$

where M_i represents the i^{th} moth, F_j is the j^{th} flame, D_i is the distance between moth *i* and flame *j*, *b* is a constant for defining the shape of the helix, and *t* is a random number between [-1, 1].

The moth-to-flame method in MFO is similar to the bubble-net feeding method in whale optimization algorithms. Mathematical formulas apply only to continuity problems. For a discrete DLBP, a three-point crossover and single-point mutation were adopted to simulate this process by Zhang et al. (2022). The detailed algorithm flow as shown in Algorithm 3 and the process is shown schematically in Fig. 7.

Algorithm 3: Moth-to-flame method

```
% Phase I: Three-point crossover
1  Select two candidate Flame and Moth.
2  Randomly generate three different natural numbers between [1, $n$ + 1], which divide
   the sequence into four parts.
3  Two different natural numbers are randomly selected between [1, 4], which
   determine the crossed segments.
4  Rearrange the elements in the cross sequences of the Moth in the order of the Flame,
   and the previous fragment is replaced with the changed fragment. Similarly, the
   same operation is performed on the segments to be crossed in the Flame.
5  Two new moths are generated.
% Phase II: Single-point mutation
6  Randomly select one mutation task between [1, $n$] and determine the position of the
   nearest immediately succeeding task and immediately preceding task of chosen one.
7  Randomly insert the chosen task to one position.
```

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

ORIGINAL UNEDITED MANUSCRIPT

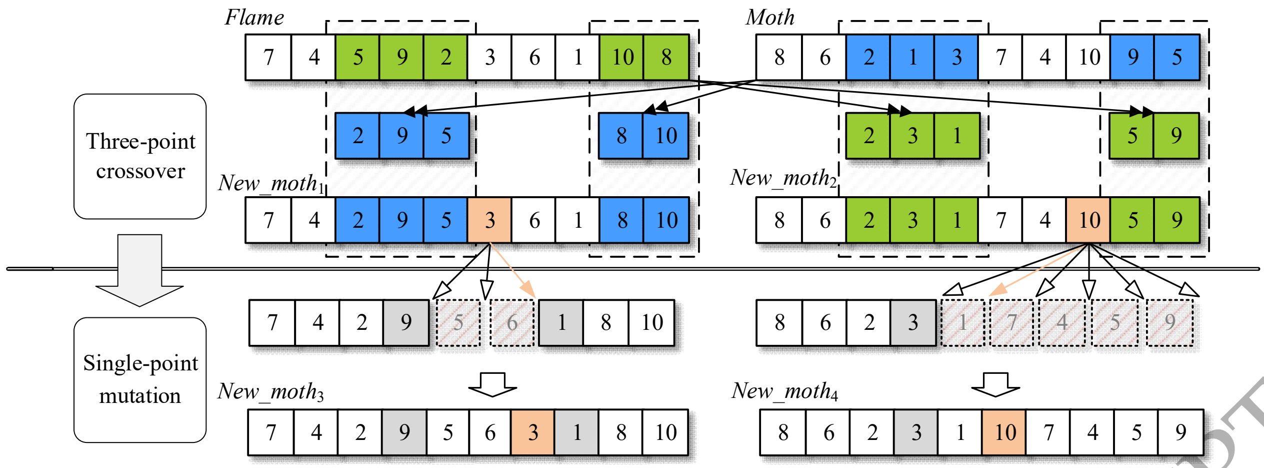


Fig. 7 Moth-to-flame method

When $N_q < N_f$(N_q indicates the size of the non-inferior set, N_f represents the number of flames), each moth randomly selects a flame and moves toward it. When $N_q \geq N_f$, each moth moves toward each flame. To prevent excessive quantity, let $N_q = n$ if the size of the non-inferior set exceeds n, where n indicates the problem size. This operation increases the optimizing speed in the early period of the algorithm and effectively improves the search performance in the late period of the algorithm.

II. Update flames

After filtering the non-inferior solutions for new moth populations generated, the results are stored in an external file (non-inferior set). The first N_f solutions with the largest crowding distance from the external files are selected as the next generation of the flame population. If $N_q < N_f$, N_Q - N_F flames are randomly generated to fill the flame population according to the method described in Section 4.1.

To ensure the convergence of the algorithm, the MFO algorithm gradually abandons flames with poor adaptability during the iteration process, and only one flame remains. The moths eventually gather near this flame. This process is described by Eq. (26).

$$N_{flame} = round\left(N_{moth} - l \cdot \frac{N_{moth} - 1}{T}\right) \tag{26}$$

where N_{flame} represents the number of flames, N_{moth} represents the number of moths, l indicates the number of iterations, and T indicates the maximum permission iterative number.

Considering the multi-objective nature of the DLBP, the real Pareto frontier consists of several non-inferior solutions. Only one flame left in the final iteration will cause the algorithm to fall into the local optimum. Therefore, we improved Eq. (26) based on the previous basis. In the initial stage of evolution, the number of flames decreases gradually with the iteration process. When the number of flames is less than that of non-inferior solutions in the external file, $N_f = N_q$. The Pareto optimal solution is retained as a new generation of the flame population. The modified formula for this process is given by Eq. (27).

$$N_{flame} = \begin{cases} round\left(N - l \cdot \dfrac{N-1}{T}\right) & N_{flame} \geq N_Q \\ N_Q & N_{flame} < N_Q \end{cases} \tag{27}$$

4.5 Improvement of MFO

The flames in the proposed MFO algorithm represent the best solution for the population. However, the moth-to-flame method may result in unwanted rapid convergence which causes the algorithm to fall into a local optimum. Therefore, a restart strategy is proposed to solve this problem. When the searching process has a trend of stagnation, the restart strategy is adopted for the flame. The non-inferior set $\boldsymbol{Q}$ of each generation is recorded; if $\boldsymbol{Q}$ remains constant for five generations, a new moth population and its corresponding new flame population are generated randomly to replace the current flame population.

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

ORIGINAL UNEDITED MANUSCRIPT

Subsequently, during the moths-to-flame phase, the location update operation is performed by the previous moths and the newly generated flame. To guarantee optimal solutions and efficient frontiers, the algorithm mixes the new population with the previous generation of the non-inferior set for Pareto screening.

5. Model verification and example analysis

All the methods were tested in a running environment with Intel (R) Core (TM) i5-9400 CPU @2.90 GHz with 16 GB RAM in Windows 10. Each algorithm was coded and run in MATLAB2018b. As described in Section 5.1, the proposed model is verified using the Gurobi exact solver. Currently, no research was conducted on parallel two-sided disassembly line problems; therefore, the proposed algorithm was evaluated using two scenarios of DLBP and TDLBP (Sections 5.2, and 5.3, respectively). These problems assumed a different form in decoding methods compared with PTDLBP in this study but were essentially the same for the optimizing process. The effectiveness of the algorithm was determined from the results described above. Finally, the proposed algorithm was used to solve PTDLBP and compared with other algorithms, including the classical algorithm for the genetic algorithm, the latest published improved whale optimization algorithm, and the unmodified MFO algorithm (Section 5.4).

5.1 Model verification

In this section, 2P8 and 2P10 examples (Kucukkoc, 2020) are combined as test examples. To verify the validity of the model and algorithm, Gurobi 9.0 was used to be solved using the MIP model developed and IMFO was used to solve the same problems simultaneously. The number of iterations was 100, and the population size was 50. The algorithm automatically ran 10 times. The results of Gurobi and a random run of the algorithm are listed in Table 1. In addition, the mean and variance of the best value of each objective in each non-inferior solution set were also recorded for 10 solving processes.

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

ORIGINAL UNEDITED MANUSCRIPT

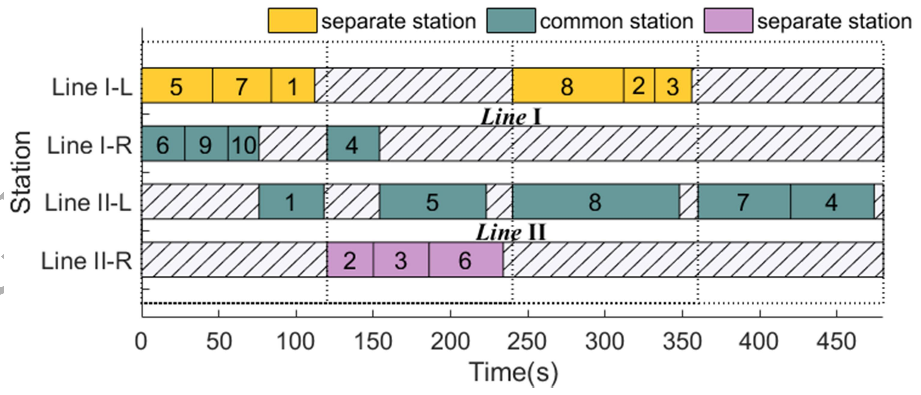
Table 1

Results of Gurobi and the IMFO algorithm for solving PTDLBP

Test problem (Line I – Line II)	CT	#	Gurobi			IMFO		
			f_1	f_2	f_3	f_1	f_2	f_3
2P8-2P10	40-40	1	**409**	-	-	411	**6604**	**40**
		2	-	**6604**	-	**409**	6924	**40**
		3	-		**40**	410	6684	**40**
		Avg.	-	-	-	**409**	**6604**	**40**
		S.D.	-	-	-	0.0	0.0	0.0
	40-60	1	**407**	-	-	410	107541	120
		2	-	**107401**	-	409	107541	148
		3	-	-	**46**	408	**107401**	166
		4	-	-	-	408	107613	**46**
		5	-	-	-	410	107561	**46**
		6	-	-	-	**407**	116361	**46**
		7	-	-	-	**407**	116281	166
		8	-	-	-	**407**	115621	286
		Avg.	-	-	-	**407**	**107401**	**46**
		S.D.	-	-	-	0.0	0.0	0.0
	60-60	1	**306**	-	-	**306**	19994	60
		2	-	**16428**	-	**306**	16470	83
		3	-	-	**23**	307	**16428**	**23**
		4	-	-	-	306	20114	**23**
		Avg.	-	-	-	**306**	**16428**	**23**
		S.D.	-	-	-	0.0	0.0	0.0

*Best results indicated in bold

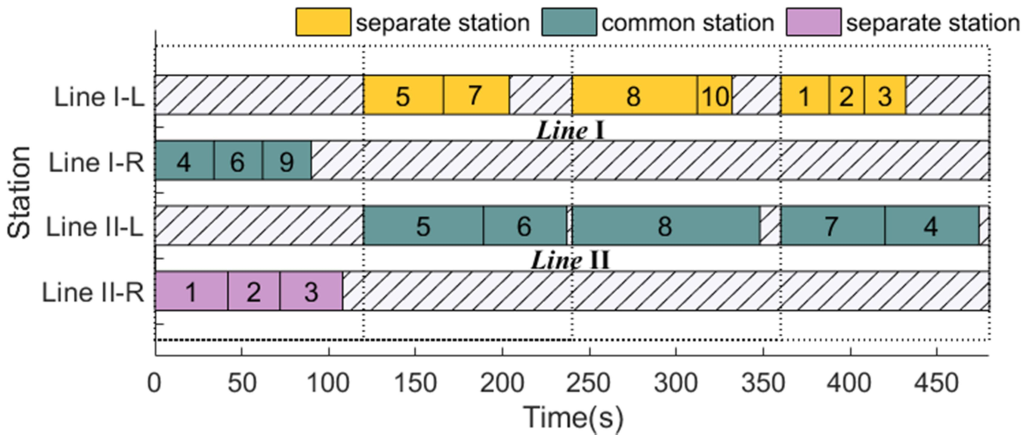
This section describes the verification of the correctness of the model. Table 1 shows that the Pareto solution set obtained by IMFO included each objective optimal value obtained by Gurobi. The means and variances of the best values indicated that each calculation result included the optimal solutions. These results demonstrated the accuracy of the model and effectiveness of the algorithm. Note that the exact solver can only obtain a single-objective optimal solution for small-scale problems, which is a method to solve the problem. However, accurately obtaining solutions in a reasonable time is a challenge for large-scale problems. In addition, DLBPs belong to multi-objective problems, and Gurobi can only obtain the optimal value of a single objective. The decision goals of the DLBP interact or even conflict with one another. There are almost no absolute optimal solutions with multiple objectives. Obtaining solutions approaching the true Pareto front is also the key to this problem. Thus, an effective multi-objective meta-heuristic algorithm was necessary for the problems considered. Fig. 8 shows an example of the corresponding scheme of the single-objective optimal solutions obtained by the algorithm, where Fig. 8a is for the objectives of the weighted line length and hazard index, and Fig. 8b is for the objective of the idle index.



(a) Minimum weighted line length $f_1 = 407$ and minimum hazard index $f_3 = 46$

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022



(b) Minimum weighted index f_2 =107401

Fig. 8 Single objective optimal solution of 2P8-2P10 with 40-60 CT

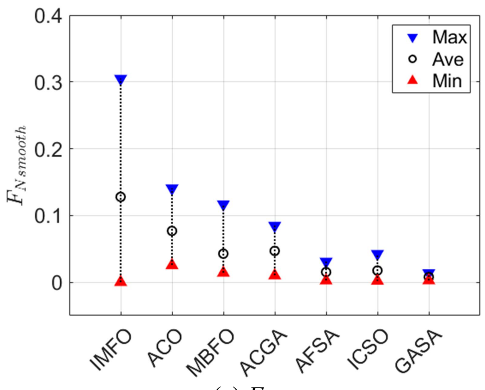
***5.2 Case study* 1**

This section solves the large-scale DLBP instance of 52 parts, and specific information was adapted from Ding et al. (2009). For P52, the optimization objectives of the problem included the unsmooth index ($F_{Nsmooth}$), idle rate (F_{idle}), and disassembly cost (F_{cost}). The results of ant colony optimization (ACO) (Ding et al., 2009), multi-objective bacterial foraging optimization (MBFO) (Yang et al., 2016), ant colony and genetic algorithm (ACGA) (Zhang et al., 2018), artificial fish swarm algorithm (AFSA) (Zhang et al., 2017), improved cat swarm optimization algorithm, (ICSO) (Zou et al., 2017), and genetic algorithm/simulated annealing (GASA) (Wang et al., 2017) are shown in Fig. 9. The parameters of compared algorithms are available in the relevant literature, and the solution results are obtained from their research. For IMFO, considering that there were only two parameters of the IMFO algorithm, i.e. the number of iterations and population size, the higher the two values, the better the result, but the longer the run time. Based on the synthesis considerations of algorithm performance and time cost, the parameter as follows: *Max_iter* = 1000, *Moth_num* = 100. The number of final output solutions is limited to 10. The proposed IMFO algorithm was run independently 10 times with an average run time of 138.2 s. The results are shown in Fig. 9, where Fig. 9a–c show the maximum, minimum, and mean values of each objective in the Pareto optimal solution set obtained by each algorithm. Except for the ACO, all F_{idle} values obtained using other algorithms were 0.0579; therefore, a 2D coordinate system was constructed with $F_{Nsmooth}$ and F_{cost} as the coordinate axes (Fig. 9d). Table 2 describes the scheme obtained using the IMFO.

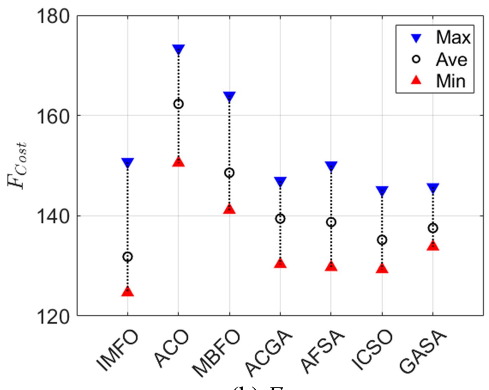
From the simulation results, the effect solution performance of the proposed IMFO could be observed. Fig. 9a—c show that the IMFO obtained the best optimum value of 0.0001 for $F_{Nsmooth}$, 124.686 for F_{cost}, and 0.0579 for F_{idle}. Note that the IMFO obtained a higher maximum and average value than the other algorithms for $F_{Nsmooth}$, which did not mean a bad convergent effect of the IMFO. This was because when the F_{idle} of schemes is the same, $F_{Nsmooth}$ is in competition with F_{cost} in the Pareto front, which means one objective improves at the expense of the other. Widely distributed solutions result in a higher average. However, this proves the diversity and wide spread of solutions to some extent. As shown in Fig. 9d, the solution obtained by IMFO completely dominated all the solutions obtained using the other algorithms. In summary, IMFO exhibited superiority in the convergence and distribution of large-scale problems for P52.

ORIGINAL UNEDITED MANUSCRIPT

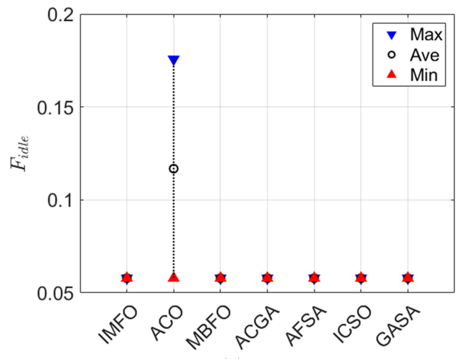
Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022



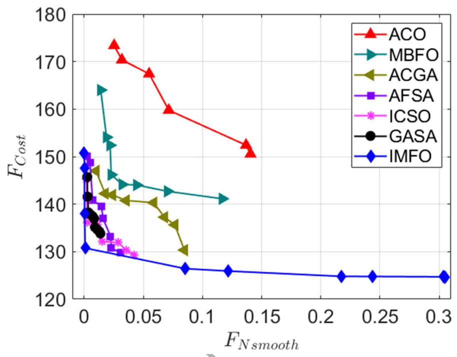
(a) $F_{Nsmooth}$



(b) F_{cost}



(c) F_{idle}



(d) Pareto frontier of seven algorithms

Fig. 9 Solution results of seven algorithms for P52

Table 2

Solutions for P52 of DLBP

No.	Disassembly sequence	F_{idle}	$F_{Nsmooth}$	F_{cost}
1	(18,21,29,33,36,3)→(2,4,14,11,28,15,31,32,37,47,46)→(1,12,23,30)→(9,25,26)→(27,34,22,35,24,42,49,51)→(7,8,13,10,16,38,17,39,40,41,43,44,45,50,48,52)→(5,6,19,20)	0.0579	0.3048	124.686
2	(18,21,29,33,36,3)→(2,4,14,11,28,15,31,32,37,47,46)→(1,12,30,42)→(9,25,26)→(23,27,34,22,35,24,49,51)→(7,8,13,10,16,38,17,39,40,41,43,44,45,50,48,52)→(5,6,19,20)	0.0579	0.3033	124.710
3	(18,21,33,36,3,47)→(2,4,28,15,29,14,11,31,32,37,46)→(1,12,23,30)→(9,25,26)→(27,34,22,35,24,42,49,51)→(7,8,13,10,16,38,17,39,40,41,43,44,45,50,48,52)→(5,6,19,20)	0.0579	0.2435	124.776
4	(18,21,33,36,3,42)→(4,28,15,29,14,11,31,32,37,47,46)→(1,12,30,34)→(9,25,26)→(2,22,23,27,35,24,49,51)→(7,8,13,10,16,38,17,39,40,41,43,44,45,50,48,52)→(5,6,19,20)	0.0579	0.2177	124.800
5	(18,21,33,36,3,42)→(4,28,15,29,14,11,31,32,16,37,47,46)→(1,12,25)→(9,26,30)→(23,27,34,22,35,24,49,51)→(2,7,8,13,10,38,17,39,40,41,43,44,45,50,48,52)→(5,6,19,20)	0.0579	0.1215	125.922
6	(4,18,28,15,29,14,31,32,42)→(21,11,33,36,3,23,16)→(1,12,47,46,30)→(9,25,26)→(27,34,22,35,24,37,49,51)→(2,7,8,13,10,38,17,39,40,41,43,44,45,50,48,52)→(5,6,19,20)	0.0579	0.0855	126.420
7	(4,21,28,15,32,33,36)→(3,2,27,29,14,31,37,42,47)→(1,25,26)→(9,12,46,30)→(18,11,16,19,23,43,44,45,49,51,40,52)→(34,22,7,35,24,8,13,10,38,41,50,48)→(5,6,17,20,39)	0.0579	0.0008	130.764
8	(21,28,15,27,33,36,47)→(3,2,4,29,14,31,32,37,42)→(1,12,25,45)→(9,18,26,46,43)→(30,11,16,19,23,49,34,51,40)→(22,7,35,24,8,13,10,38,41,44,50,48,52)→(5,6,17,20,39)	0.0579	0.0004	137.940
9	(4,26,32,33,36)→(3,18,28,15,29,14,31,37,42)→(1,2,21,25,47)→(9,12,27,45,46,43)→(30,11,16,19,23,44,49,51,40,52)→(34,22,7,35,24,8,13,10,38,41,50,48)→(5,6,17,20,39)	0.0579	0.0002	147.600
10	(4,18,28,15,29,31,32,36)→(3,14,26,37,42,33)→(1,2,21,25,47)→(9,12,27,45,46,49)→(19,11,16,23,30,34,43,51,40)→(22,7,35,24,8,13,10,38,41,44,50,48,52)→(5,6,17,20,39)	0.0579	0.0001	150.756

5.3 Case study 2

This section describes tests of the TDLBPs. Zou et al., (2018) cited a 25-scale refrigerator disassembly scenario with four optimization objectives: number of mated-stations, idle indicator, demand indicator, and hazard indicator. The comparison algorithm in this test included the bat algorithm (BA) (Zou et al., 2018) and improved differential evolution (IDE) (Xie et al.,

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

2021). Comprehensively considering the operational effect and running time comprehensively, the parameters were set as follows: *Max_iter* = 1000 and *Moth_num* = 100. No external file size was set because of the excessive number of Pareto solutions in this example. CT was set to 740 s. The algorithm ran independently 10 times. An average of 176.6 non-inferior solutions were obtained with an average runtime of 140.86 s. One of the 10 random results was selected, and eight non-inferior solutions were selected from the results. The results of the compared algorithm are taken from the corresponding literature. Detailed results are presented in Table 3.

Table 3

Solutions for 2P25 of TDLBP

	f_1	f_2	f_3	f_4	***Pareto Front***		f_1	f_2	f_3	f_4	***Pareto Front***		f_1	f_2	f_3	f_4	***Pareto Front***
BA	6	5907	461	42	**Y**	IDE	**5**	765	520	42	N	IMFO	**5**	**713**	469	33	**Y**
	6	6359	445	36	**Y**		**5**	849	500	42	N		**5**	721	486	28	**Y**
	6	8757	517	25	N		5	1495	482	37	N		6	8229	475	25	**Y**
	7	16859	454	34	N		6	7651	524	33	N		7	16509	446	26	**Y**
	7	19657	481	26	N		6	8167	479	32	N		8	27493	469	**22**	**Y**
	8	28533	485	24	N		7	16507	447	39	N		9	42003	413	29	**Y**
	9	48711	433	34	N		9	42447	429	34	N		10	60263	407	30	**Y**
	9	51679	435	30	N		9	43271	413	33	N		10	57923	**405**	35	**Y**

* Best results in bold

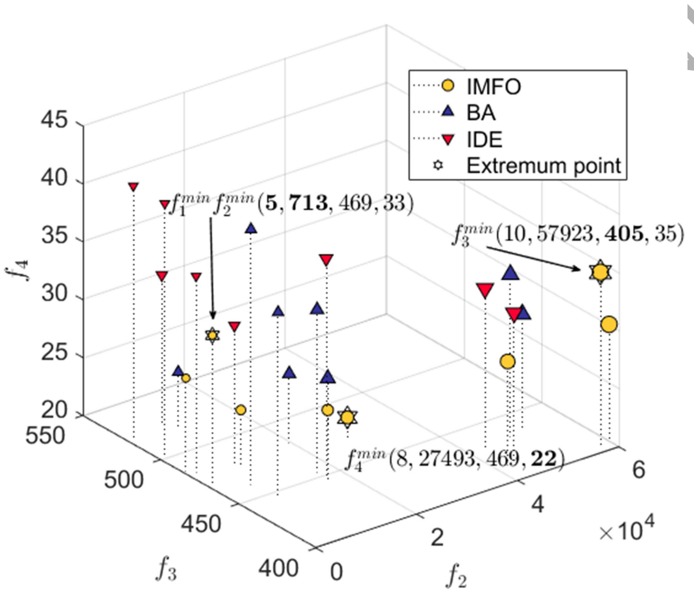


Fig. 10 Space distribution diagram of the Pareto optimal solutions of the three algorithms for 2P25

Fig. 10 depicts the space distribution of the Pareto optimal solutions of the three algorithms, where the radius size of the circle corresponds to the value of f_1. The numerical results of 2P25 proved that the IMFO algorithm provided a better value of extreme values for each target. The 24 solutions determined using the three algorithms were filtered using the Pareto method and a total of 10 solutions were obtained. All 8 solutions obtained using the IMFO belonged to the Pareto front. Two solutions computed using the BA method were non-dominated to solutions solved using the IMFO, and the schemes obtained using IDE methods were all inferior to those obtained using the IMFO. We can conclude that the proposed IMFO exhibited a better performance than the BA and IDE algorithms in two-sided disassembly line scenarios.

5.4 Experimental tests and results for the PTDLBP

This section presents the calculation results for the PTDLBP. As the PTDLBP is a new field for researchers, no directly

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

available datasets were available for this study. Therefore, we combined four two-sided examples (2p8, 2p10, 2p24, and 2p25) for the experimental test. 2p8, 2p10, and 2p25 was cited from (Kucukkoc, 2020) and 2p24 was adapted from (Liang et al., 2021). For comparison, the same problems were also solved using the unmodified MFO algorithm, improved whale optimization algorithm (IWOA) (Zhang et al., 2022) published recently, and the classical NSGA-II. The MFO had no restart strategy compared with the IMFO. To ensure fairness in comparison, the same running environment and individual update operations were adopted for the four algorithms, i.e. three-point crossover and single-point mutation. The external file capacity was set to 10, and the running time was set as the termination criterion. Through numerous tests on each algorithm, the final algorithm parameter settings and stop criterion were as follows, where the *Max_iter* is set in IMFO and MFO because of algorithm structure, which means they might have an early termination before the time limit.

Algorithms	Size	Stop criterion	Parameter
IMFO	2p8-2p10 2p10-2p10	100 s	$N = 50$, $Max\_iter = 100$
MFO			$N = 50$, $Max\_iter = 100$
IWOA			$N = 50$, $Pd = 0.8$
NSGA-II			$N = 50$, $Pc = 0.9$, $Pm = 0.3$
IMFO	2p10-2p25	300 s	$N = 80$, $Max\_iter = 300$
MFO			$N = 80$, $Max\_iter = 300$
IWOA			$N = 80$, $Pd = 0.8$
NSGA-II			$N = 80$, $Pc = 0.9$, $Pm = 0.3$
IMFO	2p24-2p24 2p24-2p25 2p25-2p25	500 s	$N = 100$, $Max\_iter = 500$
MFO			$N = 100$, $Max\_iter = 500$
IWOA			$N = 100$, $Pd = 0.8$
NSGA-II			$N = 100$, $Pc = 0.9$, $Pm = 0.3$

Table 4

Calculation results for small- and medium-size examples

Method	Problem	2p8-2p10			2p10-2p10			2p10-2p25		
	Index	1	2	3	4	5	6	7	8	9
	CT	40-40	40-60	60-60	40-40	40-60	60-60	40-40	40-60	60-60
IMFO	Avg GD	**2.17**	**22.94**	**6.66**	**8.82**	**153.97**	**32.32**	**25.77**	**1135.98**	**260.76**
	Best GD	0.00	4.61	0.00	0.00	30.47	4.87	9.08	76.61	7.43
	Avg SP	93.96	181.58	37.32	**56.61**	**2243.20**	**1339.78**	**766.40**	**5340.26**	**3020.14**
	Best SP	92.37	46.80	27.40	6.10	284.34	41.36	157.07	1400.82	0.00
MFO	Avg GD	**2.17**	30.45	8.24	15.88	266.28	64.50	72.67	1446.40	551.31
	Best GD	0.00	5.14	0.00	0.00	30.57	0.89	12.99	150.10	5.06
	Avg SP	93.96	**163.47**	**32.82**	101.08	3837.31	1379.69	822.85	5513.71	3476.34
	Best SP	92.37	60.66	30.21	31.24	283.96	39.51	119.08	919.47	0.00
IWOA	Avg GD	15.10	612.99	32.60	22.42	847.15	76.22	73.04	4850.28	519.36
	Best GD	0.00	30.00	4.50	0.00	48.10	0.61	11.45	111.44	10.05
	Avg SP	**89.85**	1646.31	80.96	111.35	2515.30	1440.78	500.53	2683.23	2194.66
	Best SP	12.88	30.11	27.40	40.68	109.60	44.59	80.70	1306.49	0.00
NSGA-II	Avg GD	328.02	2696.10	883.78	1703.56	4596.00	579.32	215.38	3165.94	610.95
	Best GD	116.57	463.13	53.55	60.65	1000.24	92.62	21.94	135.63	19.26
	Avg SP	233.17	3154.68	435.24	1837.52	6759.07	777.56	1043.54	6597.90	3274.95
	Best SP	0.00	448.87	107.49	0.00	1415.22	240.50	70.09	1300.70	0.00

*Best Avg. indicated in bold

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

Table 5

Calculation results for large-size examples

Method	Problem	2p24-2p24			2p24-2p25			2p25-2p25		
	Index	10	11	12	13	14	15	16	17	18
	CT	30-30	30-40	40-40	30-30	30-40	40-40	40-40	40-60	60-60
IMFO	Avg GD	**22.72**	634.30	24.71	**63.98**	**304.57**	**150.40**	**678.60**	**65.98**	**20.97**
	Best GD	1.53	135.71	1.09	10.69	19.25	48.55	2.63	22.73	4.63
	Avg SP	**79.42**	**651.55**	**39.59**	371.46	1807.63	435.54	**53.14**	1312.29	**50.26**
	Best SP	5.23	198.84	17.15	23.64	503.18	64.17	6.26	601.29	9.32
MFO	Avg GD	46.74	826.30	29.66	65.32	3336.87	177.34	1168.85	116.71	25.90
	Best GD	7.54	103.33	8.54	12.20	70.40	6.74	5.31	16.01	13.93
	Avg SP	108.70	3063.78	67.76	222.50	**1159.33**	**188.34**	555.11	2529.55	53.16
	Best SP	16.84	346.32	22.61	34.48	580.41	9.05	7.91	1731.52	14.56
IWOA	Avg GD	229.92	1012.49	**24.26**	75.78	1754.71	267.17	497.22	118.69	46.61
	Best GD	6.17	591.25	8.29	19.77	78.65	8.24	7.22	69.12	2.08
	Avg SP	447.92	2598.00	55.39	**101.80**	5684.16	689.62	640.76	**999.02**	84.51
	Best SP	16.34	970.87	15.99	15.07	493.33	50.63	0.00	564.66	15.49
NSGA-II	Avg GD	388.82	**251.86**	778.50	64.43	8094.29	238.85	2531.77	2477.18	382.69
	Best GD	11.91	35.40	37.82	8.25	346.66	41.59	2133.06	94.11	15.10
	Avg SP	492.13	2058.99	1093.35	416.49	6777.53	1482.19	1819.89	1430.92	1058.56
	Best SP	23.54	246.90	17.54	30.80	648.00	89.13	95.41	509.83	7.05

*Best Avg. in bold

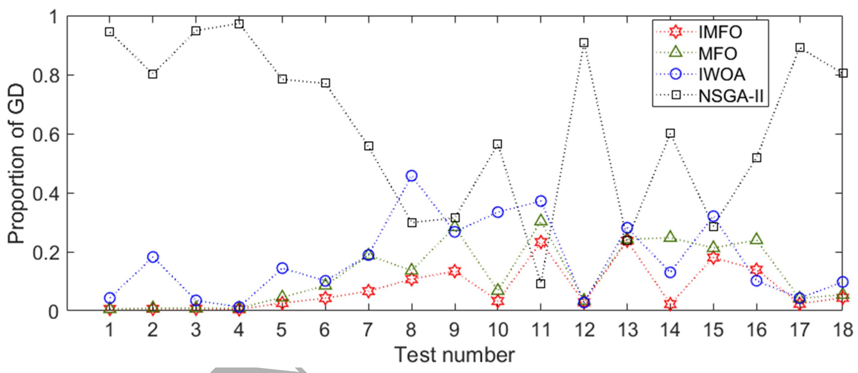


Fig. 11 Proportion of average GD obtained using four algorithms

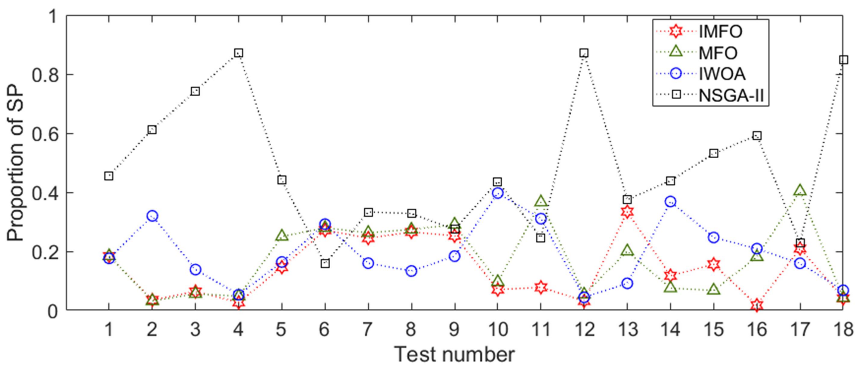


Fig. 12 Proportion of average SP obtained using four algorithms

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

To effectively evaluate the quality of non-inferior solutions to multi-objective problems, this paper introduces two different evaluation indexes to evaluate the performance of the algorithm: generational distance (*GD*) (Wang et al., 2021) and spacing (*SP*) metric (Liang et al., 2021). *GD* evaluates the convergence of the pareto set, and *SP* evaluates the diversity. All algorithms solved six scale problems with different CTs. Table 4 and Table 5 show the average and best values of the evaluation indexes obtained using each algorithm for different sizes.

As shown in Table 4, for small- and medium-scale problems, IMFO obtained the minimum average GD for each scenario than compared algorithms. For dispersibility, the proposed IMFO algorithm was superior to the other algorithms for 6 of the 9 scenarios. For large-scale problem experiments (see Table 5), the results obtained using the IMFO were better than the other three algorithms in GD and SP, where 7 of the 9 examples had optimal average GDs, and 5 of the 9 examples obtained optimal average SPs. The proportions of the average GDs and SPs obtained using the four algorithms are shown in Fig. 11 and Fig. 12, where the proportion of the average GDs is formulated by ($GD_i \Big/ \sum_{i=1}^{n} GD_i \, , i \in \{\text{IMFO}, \text{MFO}, \text{IWOA}, \text{NSGA-II}\}$), and similarly for the SPs.

In conclusion, the experimental data of PTDLBPs indicated that the proposed IMFO is significantly superior to the four compared algorithms in terms of search capability. Compared with the unmodified algorithm, the improved MFO obtained a solution set with a better GD, and the restart strategy improves the convergence performance of the MFO algorithm to a certain extent.

6. Conclusion

Two-sided disassembly lines have been extensively used in practical applications. Optimizing multiple two-sided disassembly lines effectively integrates the advantages of two-sided and parallel layouts. In this paper, a PTDLBP with fixed common stations is presented. Considering the weighted line length, line balance, and harmfulness of production lines, an MIP model is provided to describe the proposed problem. The newly constructed model is available for optimally solving small-sized instances using the Gurobi solver. An improved moth-flame algorithm is proposed to solve the PTDLBP problem. Two-stage decoding is adopted to effectively obtain the disassembly scheme, and the restart strategy improves the search capability and avoids the algorithm falling into the local optimum by replacing the optimal solution with a random solution.

The accuracy of the model and the effectiveness of the algorithm were verified using the results of small-scale scenarios using Gurobi and the IMFO method. Two classic DLBP instances were tested for comparison, and the results of the experiment demonstrated that the algorithm is reliable against DLBPs and TDLBP for different scales. For PTDLBP, several two-sided instances were combined, and a comparative study statistically confirmed the superiority of the IMFO in terms of the convergence and comprehensiveness compared with the improved whale optimization algorithm, non-dominated sorting genetic algorithm-II, and MFO algorithm without a restart strategy.

Meanwhile, in future research, parallel two-sided disassembly line problems can be expanded to different application domains, such as parallel two-sided disassembly line problems with optional common stations, complex constraints, or sequence dependence. In addition, more complex constraints such as AND/OR precedence relation and comprehensive evaluation indicators for the disassembly lines can be developed. The proposed algorithm can be applied to other combinatorial optimization problems, such as scheduling problems and freight site assignment optimization.

References

Altekin, F. T. (2017). A comparison of piecewise linear programming formulations for stochastic disassembly line balancing. *International Journal of Production Research*, *55*(24), 7412–7434. https://doi.org/10.1080/00207543.2017.1351639

Altekin, F. T., Kandiller, L., & Ozdemirel, N. E. (2008). Profit-oriented disassembly-line balancing. *International Journal of Production Research*, *46*(10), 2675–2693. https://doi.org/10.1080/00207540601137207

Bhosale, K. C., & Pawar, P. J. (2020). Production planning and scheduling problem of continuous parallel lines with demand uncertainty and different production capacities. *Journal of Computational Design and Engineering*, *7*(6), 761–774.

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

https://doi.org/10.1093/JCDE/QWAA055

Ding, L., Tan, J., Feng, Y., & Gao, Y. (2009). Multiobjective optimization for disassembly line balancing based on Pareto ant colony algorithm. *Computer Integrated Manufacturing Systems*, *15*(7), 1406–1413. https://doi.org/10.13196/j.cims.2009.07.160.dinglp.005

Edis, E. B., Sancar Edis, R., & Ilgin, M. A. (2022). Mixed integer programming approaches to partial disassembly line balancing and sequencing problem. *Computers and Operations Research*, *138*(July 2021), 105559. https://doi.org/10.1016/j.cor.2021.105559

Gökçen, H., Ağpak, K., & Benzer, R. (2006). Balancing of parallel assembly lines. *International Journal of Production Economics*, *103*(2), 600–609. https://doi.org/10.1016/j.ijpe.2005.12.001

Gungor, A., & Gupta, S. M. (2001). A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*, *39*(7), 1427–1467. https://doi.org/10.1080/00207540110052157

Gungor, A., Gupta, S. M., Pochampally, K., & Kamarthi, S. v. (2001). Complications in Disassembly Line Balancing Akiner. In S. M. Gupta (Ed.), *Proceedings of SPIE International Conference on Environmentally Concious Manufacturing* (Vol. 4193, Issue 617, pp. 289–298). https://doi.org/10.1117/12.417274

Helmi, A., & Alenany, A. (2020). An enhanced Moth-flame optimization algorithm for permutation-based problems. *Evolutionary Intelligence*, *13*(4), 741–764. https://doi.org/10.1007/s12065-020-00389-6

Hezer, S., & Kara, Y. (2015). A network-based shortest route model for parallel disassembly line balancing problem. *International Journal of Production Research*, *53*(6), 1849–1865. https://doi.org/10.1080/00207543.2014.965348

Kalayci, C. B., & Gupta, S. M. (2013). Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Expert Systems with Applications*, *40*(18), 7231–7241. https://doi.org/10.1016/j.eswa.2013.06.067

Kalayci, C. B., & Gupta, S. M. (2014). A tabu search algorithm for balancing a sequence-dependent disassembly line. *Production Planning and Control*, *25*(2), 149–160. https://doi.org/10.1080/09537287.2013.782949

Kalayci, C. B., Polat, O., & Gupta, S. M. (2015). A variable neighbourhood search algorithm for disassembly lines. *Journal of Manufacturing Technology Management*, *26*(2), 182–194. https://doi.org/10.1108/JMTM-11-2013-0168

Koc, A., Sabuncuoglu, I., & Erel, E. (2009). Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. *IIE Transactions (Institute of Industrial Engineers)*, *41*(10), 866–881. https://doi.org/10.1080/07408170802510390

Kucukkoc, I. (2020). Balancing of two-sided disassembly lines: Problem definition, MILP model and genetic algorithm approach. *Computers & Operations Research*, *124*, 1–17. https://doi.org/10.1016/j.cor.2020.105064

Kucukkoc, I., & Zhang, D. Z. (2014). Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Research*, *52*(12), 3665–3687. https://doi.org/10.1080/00207543.2013.879618

Lambert, A. J. D., & Gupta, S. M. (2008). Disassembly modeling for assembly, maintenance, reuse, and recycling. *European Journal of Operational Research*, *187*(1), 335–337. https://doi.org/10.1016/j.ejor.2007.03.022

Li, Z., & Janardhanan, M. N. (2021). Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem. *Expert Systems with Applications*, *183*(6), 1–13. https://doi.org/10.1016/j.eswa.2021.115431

Liang, J., Guo, S., Du, B., Li, Y., Guo, J., Yang, Z., & Pang, S. (2021). Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm. *Journal of Cleaner Production*, *284*, 1–20. https://doi.org/10.1016/j.jclepro.2020.125418

Liang, J., Guo, S., Zhang, Y., Liu, W., & Zhou, S. (2021). Energy-Efficient Optimization of Two-Sided Disassembly Line Balance Considering Parallel Operation and Uncertain Using Multiobjective Flatworm Algorithm. *Sustainability*, *13*(6), 1–23. https://doi.org/10.3390/su13063358

McGovern, S. M., & Gupta, S. M. (2005). Uninformed and probabilistic distributed agent combinatorial searches for the unary NP-Complete disassembly Line balancing Problem. In S. M. Gupta (Ed.), *Proc.SPIE* (Vol. 5997, pp. 59970B-59970B – 12). https://doi.org/10.1117/12.629121

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

ORIGINAL UNEDITED MANUSCRIPT

McGovern, S. M., & Gupta, S. M. (2007). Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. *International Journal of Production Research*, *45*(18–19), 4485–4511. https://doi.org/10.1080/00207540701476281

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, *89*(7), 228–249. https://doi.org/10.1016/j.knosys.2015.07.006

Mutlu, S., & Güner, B. (2021). A memetic algorithm for mixed-model two-sided disassembly line balancing problem. *28th CIRP Conference on Life Cycle Engineering*, *98*, 67–72. https://doi.org/10.1016/j.procir.2021.01.007

Özcan, U., Gökçen, H., & Toklu, B. (2010). Balancing parallel two-sided assembly lines. *International Journal of Production Research*, *48*(16), 4767–4784. https://doi.org/10.1080/00207540903074991

Özceylan, E., Kalayci, C. B., Gungor, A., & Gupta, S. M. (2019). Disassembly line balancing problem: a review of the state of the art and future directions. *International Journal of Production Research*, *57*(15–16), 4805–4827. https://doi.org/10.1080/00207543.2018.1428775

Shu, Z., Ye, Z., Zong, X., Liu, S., Zhang, D., Wang, C., & Wang, M. (2022). A modified hybrid rice optimization algorithm for solving 0-1 knapsack problem. *Applied Intelligence*, *52*(5), 5751–5769. https://doi.org/10.1007/s10489-021-02717-4

Tuncel, E., Zeid, A., & Kamarthi, S. (2014). Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning. *Journal of Intelligent Manufacturing*, *25*(4), 647–659. https://doi.org/10.1007/s10845-012-0711-0

Wang, K., Gao, L., & Li, X. (2020). A multi-objective algorithm for U-shaped disassembly line balancing with partial destructive mode. *Neural Computing and Applications*, *32*(16), 12715–12736. https://doi.org/10.1007/s00521-020-04721-0

Wang, K., Li, X., & Gao, L. (2019). A multi-objective discrete flower pollination algorithm for stochastic two-sided partial disassembly line balancing problem. *Computers & Industrial Engineering*, *130*(3), 634–649. https://doi.org/10.1016/j.cie.2019.03.017

Wang, K., Li, X., Gao, L., Li, P., & Gupta, S. M. (2021). A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Applied Soft Computing*, *107*, 1–19. https://doi.org/10.1016/j.asoc.2021.107404

Wang, K., Zhang, Z., Zhu, L., & Zou, B. (2017). Pareto genetic simulated annealing algorithm for multi-objective disassembly line balancing problem. *Computer Integrated Manufacturing Systems*, *23*(6), 1277–1285. https://doi.org/10.13196/j.cims.2017.06.013

Wu, T., Zhang, Z., Yin, T., & Zhang, Y. (2022). Multi-objective optimisation for cell-level disassembly of waste power battery modules in human-machine hybrid mode. *Waste Management*, *144*, 513–526. https://doi.org/10.1016/j.wasman.2022.04.015

Xie, M., Zhang, Z., & Jiang, J. (2021). Modeling and Optimization of Two-sided Disassembly Line Balance Problem Considering Station Constraint and Energy Consumption. *Computer Integrated Manufacturing Systems*, *27*(3), 701–715. https://doi.org/10.13196/j.cims.2021.03.005

Yang, H., Zeqiang, Z., Kaipu, W., & Lili, M. (2016). Pareto bacterial foraging algorithm for multi-target disassembly line balance problem. *Application Research of Computers*, *33*(11), 3265–3269. https://doi.org/10.3969/j.issn.1001-3695.2016.11.015

Zhang, Y., Zhang, Z., Guan, C., & Xu, P. (2022). Improved whale optimisation algorithm for two-sided disassembly line balancing problems considering part characteristic indexes. *International Journal of Production Research*, *60*(8), 2553–2571. https://doi.org/10.1080/00207543.2021.1897178

Zhang, Z., Wang, K., Zhu, L., & Cheng, W. (2018). Pareto hybrid ant colony and genetic algorithm for multi-objective U-Shaped disassembly line balancing problem. *Journal Of Southwest Jiaotong University*, *53*(3), 628–637. https://doi.org/10.3969/j.issn.0258-2724.2018.03.026

Zhang, Z., Wang, K., Zhu, L., & Wang, Y. (2017). A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem. *Expert Systems with Applications*, *86*, 165–176. https://doi.org/10.1016/j.eswa.2017.05.053

Zhu, L., Zhang, Z., & Guan, C. (2020). Multi-objective partial parallel disassembly line balancing problem using hybrid group

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022

neighbourhood search algorithm. *Journal of Manufacturing Systems*, *56*(6), 252–269. https://doi.org/10.1016/j.jmsy.2020.06.013

Zou, B., Zhang, Z., Li, L., & Cai, N. (2018). Modeling and optimization for two-sided disassembly line balancing problems. *China Mechanical Engineering*, *29*(09), 86-93+103. https://doi.org/10.3969/j.issn.1004-132X.2018.09.013

Zou, B., Zhang, Z., Li, L., & Zhu, L. (2017). Multi-objective disassembly line balancing problem based on pareto improved cat swarm optimization algorithm. *Information and Control*, *46*(4), 503–512. https://doi.org/10.13976/j.cnki.xk.2017.0503

ORIGINAL UNEDITED MANUSCRIPT

Downloaded from https://academic.oup.com/jcde/advance-article/doi/10.1093/jcde/qwac134/6901509 by guest on 17 December 2022