Optimization and Engineering (2023) 24:591–622
https://doi.org/10.1007/s11081-021-09696-y

RESEARCH ARTICLE

Models and two-phase bee algorithms for multi-objective U-shaped disassembly line balancing problem

Zixiang Li[1,2] · Ibrahim Kucukkoc[3] · Qiuhua Tang[1,2] · Zikai Zhang[1,2]

Received: 22 November 2020 / Revised: 23 September 2021 / Accepted: 22 October 2021 /
Published online: 5 November 2021

Abstract

Disassembly is the first and vital step in recycling and remanufacturing end-of-life products. Disassembly lines are utilized frequently due to high productivity and suitability. This research studies the disassembly line balancing problem on the U-shaped disassembly lines, which have higher flexibility than the traditional straight disassembly lines. A mixed-integer linear programming (MILP) model is developed to formulate the AND/OR precedence relationships with the objective of minimizing the number of stations. This model is also extended to a mixed-integer nonlinear programming model to optimize four objectives. To tackle this NP-hard problem effectively, a two-phase artificial bee colony algorithm and a bee algorithm are proposed and improved. In these algorithms, the first phase selects the stations with less loads on the last two stations for the purpose of achieving the optimal number of stations. The second phase hierarchically optimizes multiple objectives to achieve better line balances. Case studies show that the proposed MILP model obtains optimal solutions in terms of station number for the small-size instances, and the U-shaped disassembly lines obtain better fitness values than the straight disassembly lines. The comparative study demonstrates that the proposed methodologies perform competing performances in comparison with other 13 re-implemented algorithms, including tabu search algorithm, iterated local search algorithm, genetic algorithm, particle swarm optimization, three artificial bee colony algorithms and the original bee algorithm.

Keywords Disassembly line balancing · U-Shaped disassembly line · Artificial bee colony · Bee algorithm · Integer programming · Multi-objective optimization

✉ Ibrahim Kucukkoc
ikucukkoc@balikesir.edu.tr

Extended author information available on the last page of the article

1 Introduction

Sustainable development and green manufacturing are becoming the main responsibility of industry and society. Product recovery is an important process to reuse or remanufacture the valuable materials so that to minimize the amount of waste. During the product recovery process, disassembly is the first and non-ignorable step to separate end-of-life products into parts, components, or subassemblies. Due to the superiority in throughput rate and productivity over the utilization of one disassembly cell, disassembly lines are widely utilized where the products are disassembled on a set of workstations (or stations shortly) in sequence (Özceylan et al. 2018).

The disassembly line balancing problem (DLBP) aims at allocating disassembly tasks to workstations across a disassembly line with one or several optimization criteria while satisfying the precedence relationship and cycle time constraints (Güngör and Gupta 1999). Precedence relationship constraint ensures that a part can be removed only when its predecessors have been removed; cycle time constraint indicates that the total operation time (or workload) of a workstation is not larger than the cycle time given (Kucukkoc 2020).

Güngör and Gupta (1999) presented the pioneering work on DLBP and later Gungor and Gupta (2001) proposed a shortest-path formulation to solve a DLBP with task failures. Since then, DLBP has been an active research area and many heuristics and exact methods have been applied. Güngör and Gupta (2002) utilized a heuristic method to study DLBPs in different situations, and McGovern and Gupta (2003) developed a greedy/2-opt hybrid algorithm to solve multi-objective DLBP. Another 2-opt algorithm was also utilized by Ren et al. (2018a) along with the weights-based multi-criteria decision process. Recently, Mete et al. (2016) developed an effective beam search heuristic to optimize the number of stations. Regarding the exact methods, Li et al. (2020) and Li et al. (2019c) developed the branch, bound and remember algorithms to optimize the number of stations. There also are many studies on mixed-integer linear programming (MILP) models (Altekin and Akkan 2012; Altekin et al. 2008; Koc et al. 2009; Kucukkoc et al. 2020; Liu et al. 2019; Mete et al. 2018; Paksoy et al. 2013) to formulate the problem mathematically. However, MILP model is weak in solving large-size instances due to excessive computational time requirements.

As DLBP is NP-hard (McGovern and Gupta 2007a, b) and multiple objectives are involved, metaheuristics have been applied to obtain near optimal solutions within a considerably less amount of computational time. McGovern and Gupta (2007a, b) proposed a genetic algorithm to solve the multi-objective DLBP and Kalayci et al. (2016) hybridized it with a variable neighborhood search method to tackle the sequence-dependent DLBP. There are much more algorithms in literature, including genetic algorithm (Kalayci et al. 2016; McGovern and Gupta 2007a, b; Zhang et al. 2019), ant colony optimization algorithm (Agrawal and Tiwari 2008; Çil et al. 2020; Ding et al. 2010; Kalayci and Gupta 2013a; McGovern and Gupta 2006), simulated annealing algorithm (Fang et al. 2020; Wang et al. 2019a), tabu search algorithm (Kalayci and Gupta 2014), artificial bee colony algorithm (Kalayci and Gupta 2013b; Kalayci et al. 2015; Liu and Wang 2017; Wang et al. 2019a, b), bees algorithm (Liu et al. 2020, 2018), particle swarm optimization algorithm (Kalayci and Gupta 2013c; Xiao et al. 2017), variable neighborhood search algorithm (Ren et al. 2018b), iterated local search method (Li et al. 2019b) and other evolutionary algorithms (Fang et al. 2019; Ren et al. 2017; Wang 2019b; Yang et al. 2019; Zhang et al. 2017; Zhu et al. 2018). A comprehensive review of the applied algorithms refers to Özceylan et al. (2018), Deniz and Ozcelik (2019) and Gao et al. (2020).

As far as the line layout involved, the majority of previous studies consider the DLBP on the straight disassembly line, whereas the studies on the U-shaped disassembly line balancing problem (UDLBP) are limited. Agrawal and Tiwari (2008) studied the mixed-model U-shaped disassembly line balancing and sequencing problem. Later on, Li et al. (2019b) addressed the sequence-dependent UDLBP with iterated local search method. Recently, Wang et al. (2020) considered the UDLBP with partial destructive mode and developed a multi-objective discrete flower pollination algorithm.

From the above literature review, it is observed that the traditional UDLBP is not studied whereas the sequence-dependent UDLBP and UDLBP with partial destructive mode are studied. Only few algorithms have been developed and it is necessary to develop more efficient algorithms. To fill the above gaps, this study considers the multi-objective UDLBP and develops two mathematical models and two improved metaheuristics. Notice that, this study differentiates from the published papers in both the problem setting and solution methodology. For instance, Wang et al. (2020) only considered the AND precedence relationships and this study, as will be seen in the following sections, considers the more complex AND/OR precedence relationships. Wang et al. (2020) proposed Pareto multi-objective algorithm and this study presents two-phase bee algorithms. The main contributions are presented as follows.

(1) Two mathematical models are developed to formulate the UDLBP: the first is a MILP model to minimize the number of stations and the second is a mixed-integer nonlinear programming model (MINLP) to optimize four objectives. Different from the model presented by Wang et al. (2020), the developed models utilize new variables and expressions to formulate the precedence relationships. The developed two models can tackle the AND/OR precedence relationships, whereas the model in Wang et al. (2020) only considers the AND precedence relationships.

(2) This study develops and improves two-phase artificial bee colony algorithm (TABC) and two-phase bee algorithm (TBA) to tackle the UDLBP. The first phase selects the stations with less loads on the last two stations for the purpose of achieving the optimal number of stations; the second phase optimizes the multiple objectives with the hierarchy method to achieve better line balances. The proposed TABC is enhanced utilizing a new onlooker phase to remove poor-quality solutions, and a new scout phase to achieve a high-quality diversified swarm. It also optionally employs a local search mechanism to enhance the

exploitation capacity. The proposed TBA is enhanced utilizing a new employed bee phase to accelerate the evolution process and a new scout phase to achieve a high-quality diversified swarm. All these improvements are in favor of TABC and TBA to achieve a proper balance between exploration and exploitation.

(3) This study conducts a comprehensive comparative study to evaluate the developed models and algorithms. Case studies demonstrate that the proposed MILP model obtains optimal solutions regarding the number of stations for the small-size instances, and the U-shaped layout obtains a better line balance than the straight disassembly line. Computational study demonstrates that the improvements enhance the TABC and TBA by a significant margin. TABC and TBA achieve a competing performance in comparison with 13 algorithms, including tabu search algorithm, iterated local search algorithm, genetic algorithm, two particle swarm optimization algorithms, teaching–learning-based optimization algorithm, two cuckoo search algorithms, improved migrating birds optimization, three artificial bee colony algorithms and the original bee algorithm.

The remainder of this study is organized as follows. Section 2 presents an example to highlight the advantages of the U-shaped disassembly line layout. Section 3 describes the UDLBP and presents the mathematical formulations. Subsequently, the proposed TABC and TBA are introduced in Sect. 4. The computational study is presented in Sect. 5 and the conclusions and future research venues are given in Sect. 6.

2 Motivation example

U-shaped lines contain entrance side and exit side and one station on a U-shaped line is divided into two sub-stations: sub-station on the entrance side and sub-station on the exit side. The main advantage of the U-shaped lines over straight lines is that U-shaped lines have higher flexibility and many more possible allocations of tasks, which help obtain highly efficient line balances. An example is provided to highlight the advantages of the U-shaped lines. Figure 1 presents the precedence relationships diagram with eight parts to be removed.

Table 1 presents the optimal task assignment of these tasks on a straight line and a U-shaped line in terms of the number of stations with a cycle time of 20 time-units. As seen in Table 1, five stations are utilized on the straight line. However, only four stations are required to perform the same jobs on a U-shaped disassembly line. The operator on station 1 performs tasks 4, 6 and 8 on the exit side while the worker

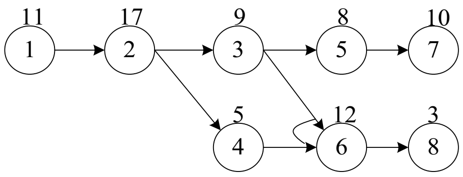

Fig. 1 Precedence diagram of the illustrated example with eight parts

Table 1 The optimal task assignment on the straight line and U-shaped line

	Stations	Station 1	Station 2	Station 3	Station 4	Station 5
Straight line	Tasks	1	2	3,5	7,4	6,8
	Task operation times	11	17	9,8	10,5	12,3
	Total workload	11	17	17	15	15
	Line balance	$(20-11)^2+(20-17)^2+(20-17)^2+(20-15)^2+(20-15)^2=149$				
U-shaped line	Tasks on the entrance side	–	–	1	–	
	Tasks on the exit side	8,6,4	7,5	3	2	
	Task operation times on the entrance side	–	–	11	–	
	Task operation times on the exit side	3,12,5	10,8	9	17	
	Total workload	20	18	20	17	
	Line balance	$(20-20)^2+(18-20)^2+(20-20)^2+(17-20)^2=13$				

on station 3 first operates task 1 on the entrance side and later operates task 3 on the exit side. One clear advantage of a U-shaped disassembly line is the possibility of consuming fewer number of workstations (and so workers).

Regarding the line balance, it is calculated with $\sum_{m=1}^{M}\left(CT-T_m\right)^2$, where CT is the cycle time, T_m is the total operation time (workload) on station m, and M is the number of stations. It is observed that the line balance (or fitness) of the straight line is 149 whereas the line balance of the U-shaped line is only 13. Clearly, U-shaped disassembly line has a better line balance with a smoother idle time distribution. In short, the U-shaped layout consumes fewer workers and obtains a better line balance even for this small-sized example.

3 Problem statement and formulation

This section first describes the considered UDLBP and later presents the mathematical formulations proposed.

3.1 Problem description

The precedence relationships in DLBP are much more complex than that in the assembly line balancing problem (ALBP). There are only AND precedence relationships in ALBP, whereas there are AND precedence, OR precedence and complex AND/OR precedence relationships in DLBP (Güngör and Gupta 2002). For simplicity, predecessors in AND precedence relationship are referred to as AND predecessors; the predecessors in OR precedence relationship are referred to as OR predecessors. One part can be removed only when all its AND predecessors have been removed; one part can be removed when at least one of its OR predecessors has been removed. Figure 2 presents the example precedence relationships diagram with

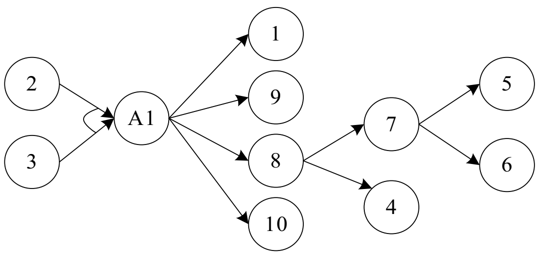

Fig. 2 Precedence diagram of an example

10 parts in the nodes and one dummy part, namely A1 (it is to simplify the presentation). The real parts need positive removal times, whereas the dummy task does not consume any time. Here, OR precedence tasks are linked via an arc, e.g. task 2 and task 3 are OR predecessors of the dummy task A1, and task A1 can be removed when either task 2 or task 3 has been removed.

Figure 3 presents two layouts of the disassembly lines: straight disassembly lines and U-shaped disassembly lines. In the straight line, one worker performs the tasks in one station. Nevertheless, the U-shaped disassembly line is divided into entrance side and the exit side and a station is divided into two sub-stations: sub-station on the entrance side and the sub-station on the exit side. A worker in the U-shaped line operates the removal tasks on the entrance side and later on the exit side, and then comes back to the entrance side again. One clear superiority of the U-shaped line is that it usually has higher flexibility and might require fewer workers (or stations).

The studied UDLBP can be described as allocating the tasks in the precedence relationships diagram to a set of stations with one or several optimization criteria. There are two main constraints to be satisfied: the cycle time constraint and precedence relationship constraint. Cycle time constraint requires that the total operation times on the entrance side or the exit side of every station is less than or equal to

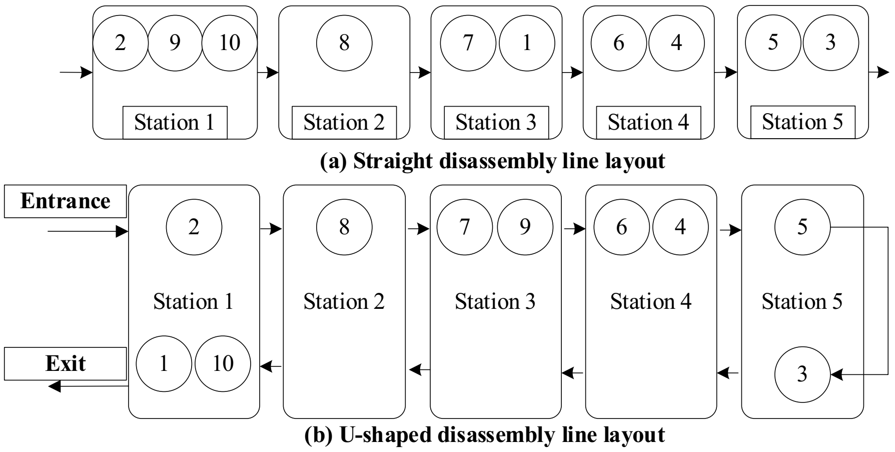

Fig. 3 Layout of (**a**) straight disassembly line and (**b**) U-shaped disassembly line

the given cycle time. Precedence relationship constraint requires that a part can be removed when all its AND predecessors or at least one OR predecessor has been removed.

3.2 Mathematical models

This section presents two new models over the one presented by Wang et al. (2020). The model presented by Wang et al. (2020) is referred to as Model-1 (provided in "Appendix A"). One clear drawback of Model 1 is that it cannot handle the OR precedence relationships. Hence, this study develops two new models to deal with the AND/OR precedence relationships. The newly developed models are denoted with Model-2 and Model-3. Model-2 is the MILP model, which aims to minimize the number of stations and uses the same expressions with Model-1 to handle the precedence relationship constraints. Model-3 is the MINLP model, which minimizes four objectives. The notations utilized in the models are given as follows.

Indices	
i, j, h	Task/part index, $i, j, h \in \{1, 2, \dots, N\}$, where N is the number of real and dummy tasks/parts
m, n	Sub-station index, $m, n \in \{1, 2, \dots, 2M\}$, where M is the maximum number of stations allowed to be opened
CT	Cycle time
t_i	Operation time of performing task i (or removal time of part i)
ANDP(i)	Set of AND predecessors of task i
ORP(i)	Set of OR predecessors of task i
ORPT	Set of tasks which have OR predecessors
h_i	1, if part i is hazardous; 0, otherwise
d_i	Demand; quantity of part i requested
Decision variables	
T_m	The total operation time on station m
$x_{i,m}$	1, if task/part i is allocated to the mth sub-workstation; 0, otherwise
z_m	1, if station m is opened; 0, otherwise
w_{ij}	1, if task i is executed before task j; 0, otherwise
s_i	Sequence of task i in the solution

Model 2, is formulated with Eq. (1), constraints (5–8) and constraint (17) while Model 3 is composed of Eqs. (1–17). Specifically, the objective function given in Eq. (1) minimizes the number of stations. Equation (2) aims to optimize the idle time distribution while Eq. (3) tries to ensure that hazardous parts are removed first. The last objective function given in Eq. (4) indicates that a part with larger demand should be removed first. Constraint (5) handles the task assignment and constraint (6) handles the cycle time constraint. Constraints (7) and (8) deal with AND precedence relationships and OR precedence relationships; respectively. Constraint (7) guarantees that a task can be allocated when all its AND predecessors have been assigned to former sub-stations or former positions of the same sub-station. Constraint (8) ensures that a task can be allocated when at least one of its OR predecessors has been assigned to a former sub-station or a former position of the same sub-station. Constraint (9) calculates the value of T_m, which is the total operation time on station m. Constraints (10–15) determine the value of w_{ij}, where w_{ij} is equal to 1 when task i is allocated to the former station or operated before task j on the same station. Constraint (16) calculates the sequence of task i in the solution. Finally, constraint (17) restricts the value of the decision variables.

Recall that Model 2 is an MILP model and it can solve the small-size instances optimally. Model 3 is a mixed-integer programming model and it has four objectives, where objective f_2 is nonlinear. Model 3 can be tackled with the MINLP solver by combining the objectives into one weighted objective.

$$\text{Min } f_1 = \sum_{m=1}^{M} z_m \tag{1}$$

$$\text{Min} f_2 = \sum_{m=1}^{M} z_m \cdot \left(CT - T_m\right)^2 \tag{2}$$

$$\text{Min } f_3 = \sum_{i=1}^{N} \left(s_i \cdot h_i\right) \tag{3}$$

$$\text{Min} f_4 = \sum_{i=1}^{N} \left(s_i \cdot d_i\right) \tag{4}$$

$$\sum_{m}^{M} \left(x_{im} + x_{i,2M+1-m}\right) = 1 \quad \forall i \tag{5}$$

$$\sum_{i=1}^{N} t_i \cdot \left(x_{im} + x_{i,2M+1-m}\right) \le z_m \cdot CT \quad \forall m \tag{6}$$

$$x_{im} \le \sum_{n=1}^{m} x_{jn} \quad \forall m, i, j \in \text{ANDP}(i) \tag{7}$$

$$x_{im} \le \sum_{j \in ORP(i)} \sum_{n=1}^{m} x_{jn} \quad \forall m, i \in \text{ORPT} \tag{8}$$

$$\sum_{i=1}^{N} t_i \cdot \left(x_{im} + x_{i,2M+1-m}\right) = T_m \quad \forall i \tag{9}$$

$$w_{ii} = 0 \quad \forall i \tag{10}$$

$$w_{ij} + w_{ji} = 1 \quad \forall i, j \text{ and } i < j \tag{11}$$

$$w_{ji} = 1 \quad \forall j \in \text{ANDP}(i) \tag{12}$$

$$\sum_{j \in \text{ORP}(i)} w_{ji} \geq 1 \quad \forall i \in \text{ORPT} \tag{13}$$

$$x_{im} + x_{jn} - 1 \leq w_{ij} \quad \forall i, j, m, n, m < n \tag{14}$$

$$w_{ih} + w_{hj} - 1 \leq w_{ij} \quad \forall i, j, h, i \neq h, h \neq j \text{ and } i \neq j \tag{15}$$

$$s_i = N - \sum_{j=1}^{N} \left(w_{ij}\right) \quad \forall i \tag{16}$$

$$z_m \in \{0, 1\} \ \forall m; x_{im} \in \{0, 1\} \ \forall i, m; w_{ij} \in \{0, 1\} \ \forall i, j \tag{17}$$

The necessity of the newly developed models together with the main differences between the published model (i.e. Model-1) and the formulated models (Model 2 and Model 3) are explained as follows.

(1) Model 1 cannot tackle the OR precedence relationships, whereas Model 2 and Model 3 are capable of tackling the OR precedence relationships with the help of Constraint (8). Model 2 and Model 3 utilize different indices and decision variables from Model 1, where one station is divided into two sub-stations and the sub-stations are encoded with 1, 2,…, M,…, 2 M (see Fig. 4). After this transition, the precedence relationships can be properly handled with the similar method in DLBPs.

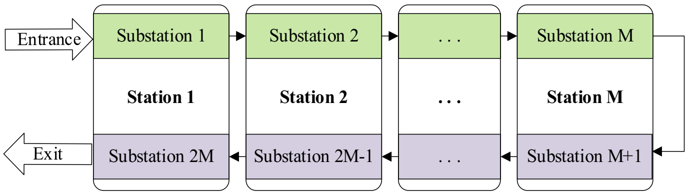

Fig. 4 The encoding indexes of the sub-stations

(2) Model 3 is a MINLP model which formulates the objectives with mixed-integer programming method. Nevertheless, most published studies for DLBP (Ding et al. 2010; McGovern and Gupta 2007a, b; Wang et al. 2020; Xiao et al. 2017) only describe the objective functions. However, Model 3 contains the objective formulations and it can solve the small-size instances utilizing the MINLP solver.

4 Proposed two-phase algorithms

As the considered UDLBP is a multi-objective global optimization problem of type NP-hard (Özceylan et al. 2018), the current exact methods cannot solve the large-size instances in an acceptable amount of computation time. Hence, this section selects and improves two powerful metaheuristics, artificial bee colony algorithm (ABC) and bee algorithm (BA). They are selected as they have obtained competing performance in the DLBPs and ALBPs (Kalayci and Gupta 2013b; Kalayci et al. 2015; Kucukkoc et al. (2019); Liu et al. 2020, 2018; Wang et al. 2019a, b, c) and they can be applied to the considered discrete problem directly with neighbor operators. Meanwhile, the proposed two-phase algorithms, namely TABC and TBA, utilizes several improvements to obtain a proper balance between exploration and exploitation. Especially, they utilize a two-phase evaluation mechanism to attempt to achieve the optimal number of stations. The first phase selects the stations with less workloads on the last two stations for the purpose of achieving the optimal number of stations; the second phase optimizes the multiple objectives with the hierarchy method to achieve a better line balance.

4.1 Two-phase evaluation

In the two-phase algorithms (TABC and TBA) as presented in Algorithm 1, Phase I terminates when the optimal number of stations is achieved (equal to the lower bound at the root) or the half of the time limit is reached. After the termination of Phase I, Phase II is conducted until the computation time reaches the pre-determined time limit.

In Phase I, the algorithm minimizes the objective formulated by $ns + \left(T_{ns-1} + T_{ns}\right)/(2 \cdot CT)$ for the purpose of achieving the optimal number of stations, where T_{ns-1} and T_{ns} are station workloads on the last two workstations. In Phase II, the algorithm aims to hierarchically minimize the objective functions given in Sect. 3.2 with the *hierarchy method* employed in McGovern and Gupta (2006) and Kalayci and Gupta (2013b). Namely,f_1 is first evaluated, f_2 takes effect when individuals have the same values of f_1, and finally f_4 is considered only when individuals have the same values of f_1,f_2 and f_3.

Algorithm 1: The main procedure of two-phase algorithms

```
Initialize PS individuals in the swarm; %PS indicates population size
Phase I    While (the optimal solution is not achieved and the computation time is less than half of
           the time limit)
               % Minimizing the objective of $ns + (T_{ns-1} + T_{ns})/(2 \cdot CT)$ for the purpose of
               achieving the optimal number of stations.
               Update the population;
           Endwhile
Phase II   While (the pre-determined time limit is not exceeded)
               % Minimizing the objectives in Section 3.2 with the hierarchy method to achieve
               a better line balance.
               Update the population;
           Endwhile
Output: The best individual
```

The main critical question about the two-phase evaluation might be whether it is necessary to utilize the two phases. The main reasons of utilizing Phase I lie behind that the optimization of the line balance (f_2) might be an obstacle of achieving solutions with smaller number of stations when solving the 'hard' instances (Li et al. 2019b). Suppose that there are two solutions with the same number of stations: one is well-balanced and the other one has a worse workload distribution (due to a little workload on the last workstation). The well-balanced solution is very difficult or even impossible to be transferred into a solution with the smaller number of stations. However, the other solution has a large possibility of reducing one station with small modifications. Hence, Phase I preserves the solutions with less station loads on the last two stations and increases the possibility of reducing one station. If Phase I is not utilized, the solutions with better line balances, which refer to the secondary objective, will be preserved. The utilization of Phase I will be tested in Sect. 5.2. As you will see, the utilization of Phase I is capable of achieving smaller number of stations for the 'hard' instances.

4.2 Encoding, decoding and neighbor operators

This research utilizes the task permutation for encoding, where all the tasks are listed in sequence. The tasks in former positions of the task permutation have higher priorities and are allocated first. Figure 5 illustrates the encoding scheme for the illustrated example given in Sect. 3.1. Here, task 1 is in the first position and thus it has the highest priority to be allocated.

To transfer the task permutation into a feasible solution, a decoding scheme (see Algorithm 2) is necessary. The detailed task assignment based on a task permutation with a cycle time of 40 is illustrated in Fig. 5, where the input data for this instance are presented in Table 2. Specifically, when allocating the first task, there are eight

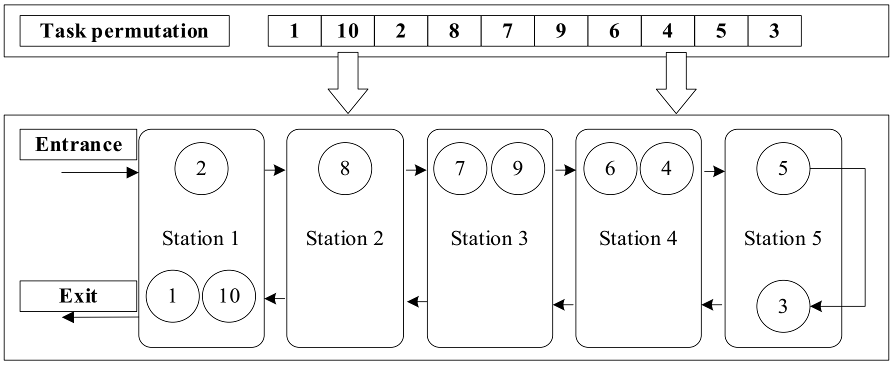

Fig. 5 Representation of the solution for the illustrated example

Table 2 Knowledge database for the illustrated example

Task	Part removal time	Hazardous	Demand
1	14	No	0
2	10	No	500
3	12	No	0
4	18	No	0
5	23	No	0
6	16	No	485
7	20	Yes	295
8	36	No	0
9	14	No	360
10	10	No	0

tasks that can be allocated (satisfying the precedence relationship constraints): tasks 2 and 3 to the entrance side and tasks 1, 4, 5, 6, 9 and 10 to the exit side. As task 1 has the highest priority (existing first in the task permutation), task 1 is allocated at first to the exit side of station 1.

Algorithm 2: Decoding procedure of UDLBP

```
Step 1: If all tasks are assigned, terminate; otherwise, execute step 2;
Step 2: Open a new station;
Step 3: Obtain the assignable task set based on the cycle time and precedence relationship constraints;
        % A task is assignable if it can be allocated to either entrance side or exit side without the
        violation of the precedence relationship constraints and can be finished within the cycle time
        given.
Step 4: If assignable task set is not empty, execute Step 5; otherwise, go to Step 1.
Step 5: Select one assignable task in the former position of the task permutation and update the
        remaining capacity of the current station; go back to Step 3.
```

Clearly, $f_1 = 5$ as five stations are utilized in Fig. 5. Regarding the value of f_2, the total operation times on five stations are [10 + 14 + 10, 36, 20 + 14, 16 + 18, 23 + 12] and thus the idle times on five stations are [6, 4, 6, 6, 5]. Subsequently, the value of f_2 is calculated as $6^2 + 4^2 + 6^2 + 6^2 + 5^2 = 149$. As for f_3 and f_4, the solution sequence or the task operation sequence is necessary in the calculation. In UDLBP, the task operation sequence is not the task permutation in decoding, and it is [2, 8, 7, 9, 6, 4, 5, 3, 10, 1] in Fig. 5 as the tasks on the former sub-stations are operated first. Afterwards, on the basis of the task operation sequence the values for f_3 and f_4 can be calculated as follows: $f_3 = 3(= 1 \times 0 + 2 \times 0 + 3 \times 1 + 4 \times 0 + 5 \times 0 + 6 \times 0 + 7 \times 0 + 8 \times 0 + 9 \times 0 + 10 \times 0)$; $f_4 = 5250(= 1 \times 500 + 2 \times 0 + 3 \times 295 + 4 \times 360 + 5 \times 485 + 6 \times 0 + 7 \times 0 + 8 \times 0 + 9 \times 0 + 10 \times 0)$.

Task permutations are modified through swap and insert operators to have high-quality neighbor solutions (Kalayci and Gupta 2013b). Swap operator exchanges the positions of two different tasks in the task permutation while insert operator inserts one task into a different position in the task permutation. Figure 6 illustrates the example applications of these two operators. To increase the search space and diversify the population, both operators are utilized and one of them is randomly selected (with equal probability) to modify the individual and obtain a new neighbor solution.

4.3 Procedure of the original ABC

The main procedure of the original ABC, referred to as OABC hereafter, is illustrated in Algorithm 3. The original ABC starts with initializing a set of solutions, referred to as food sources. Afterwards, employed bee phase, onlooker phase and scout phase are conducted in sequence and this main loop is repeated until a termination criterion is satisfied. In employed bee phase, employed bees explore the neighbor of the incumbent food sources. An onlooker explores promising food sources determined on the basis of the selection probability utilizing a roulette wheel selection. A scout performs a random

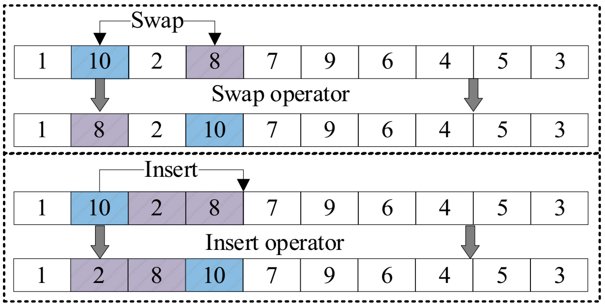

Fig. 6 Swap and insert operators

search to find a new food source when no further improvement can be achieved. In this algorithm, employed bees and onlookers perform exploitation, whereas the scout performs exploration.

```
Algorithm 3: The main procedure of the OABC
% Parameter PS is population size, and parameter limit indicates that one individual should
be abandoned when it has not been improved for limit times consecutively.
-----------------------------------------------------------------
Initialize PS individuals in the swarm;
While (Termination criterion is not met)
    For p:=1 to PS        % Employed bee phase
        Achieve a neighbor solution $p'$ of individual $p$;
        $p \leftarrow p'$ when $p'$ obtains a better fitness value;
    Endfor
    Obtain the selection probabilities of individuals;
    For p:=1 to PS        % Onlooker phase
        Select one individual, namely s, with roulette wheel selection;
        Achieve a neighbor solution $s'$ of individual s;
        $s \leftarrow s'$ when $s'$ obtains a better fitness value;
    Endfor
    % Scout phase
    Select one individual that has not been improved for limit times consecutively;
    Replace the selected individual with a new random individual;
Endwhile
-----------------------------------------------------------------
Output: The best individual
```

4.4 Procedure of the improved TABC

Following the main procedure of the OABC, the main procedure of improved TABC with Phase I and Phase II is illustrated in Algorithm 4. In the improved TABC, r is a new parameter within $[0, 1]$ and Rand$[0, 1]$ is a random number within $[0, 1]$. As you can see, the improved TABC makes modifications on employed bee phase, onlooker phase, scout phase and utilizes local search. The main segments and modifications are further explained below.

```
Algorithm 4: The main procedure of improved TABC
% Parameter $r$ means the probability to conduct the local search.
Initialize PS individuals in the swarm;
While (Termination criterion is not met)
    For $p$:=1 to PS        % Modified employed bee phase
        Achieve a neighbor solution $p'$ of individual $p$;
        $p \leftarrow p'$ when $p'$ obtains a better or the same fitness value;
    Endfor
    For $p$:=1 to PS        % Modified onlooker phase
        Select one individual $\mathrm{s}$ with binary tournament selection;
        Achieve a neighbor solution $\mathrm{s}'$ of individual $\mathrm{s}$;
    Endfor
    Select the best and non-duplicated PS solutions from the incumbent solutions and new
    neighbor solutions as the new swarm;
    % Modified scout phase
    For any individual $p$ which is a duplicated solution or has not been improved for
    consecutive limit times
        Achieve a set of neighbor solutions of individual $p$ after applying neighbor
        operators for several times;
        Select the best neighbor solution $p'$ (different from the $p$) to replace solution $p$;
    Endfor
    For any individual $p$ under $\mathrm{Rand}[0,1] \le r$   % Local search
        For $k$:=1 to $N$ do
            Achieve a neighbor solution $p'$ of individual $p$;
            $p \leftarrow p'$ when $p'$ obtains a better or the same fitness value;
        Endfor
    Endfor
Endwhile
Output: The best individual
```

The modified employed bee phase utilizes a new acceptance criterion, where $p \leftarrow p'$ when p' obtains a better or the same fitness value. In the original employed bee phase, $p \leftarrow p'$ when p' obtains a better value. In the modified employed bee phase, the incumbent individual is also replaced when the same objective value is obtained as there are many different task assignments having the same objective value. This modification somewhat enhances the exploration capacity. The modified onlooker phase selects one individual s with binary tournament selection and later achieves a neighbor solution s' of solution s with neighbor operators. After generating a number of PS individuals, the best and non-duplicated PS solutions are selected from the incumbent solutions and new neighbor solutions are designated as the new swarm. This modification accelerates the search process by replacing the poor-quality solutions with high-quality solutions.

The modified scout phase abandons duplicated solutions or the solution which has not been improved for *limit* consecutive times. This abandoned solution is replaced with the best neighbor solution from a set of candidates generated after applying neighbor operators for several times. In the original employed bee phase, on the contrary, a randomly generated solution replaces the abandoned individuals, whereas the randomly generated solution usually has a poor performance. In short, this modification utilizes the high-quality neighbor solutions to replace the abandoned ones to achieve a high-quality swarm with diversity. Local search is applied with a probability for any individual. Local search is not executed when r is set to 0.0; it is executed on each individual when r is set to 1.0. The utilization of local search aims at emphasizing exploitation. All in all, the modifications help the improved TABC to achieve a proper balance between exploitation and exploration.

4.5 Procedure of the original BA

The main procedure of the original BA, referred to as OBA, is provided in Algorithm 5. OBA initializes PS scout bees at first, and the main loop is repeated until the termination criterion is met. In this main loop, the scout bees are first sorted based on the objective values and so the best solution should be in the first position. For any individual in the best employed bees, this algorithm achieves a set of *nep* neighbor solutions of individual p and replaces the incumbent solution with the best neighbor solution p' when p' obtains a better fitness value. Similarly, for any individual in the remaining employed bees, this algorithm achieves a set of *nsp* neighbor solutions of individual p and replaces the incumbent solution with the best neighbor solution p' when p' obtains a better fitness value. Finally, the remaining $PS - P$ individuals are replaced with new randomly generated individuals. OBA emphasizes exploitation in updating the employed bees and emphasizes exploration in updating scouts.

Algorithm 5: The main procedure of the OBA

% Parameter *s* is the number of scouts, parameter *P* is the number of employed bees, parameter *e* is the number of best-employed bees, parameter *nep* is the number of bees for the best *e* individuals, and parameter *nsp* is the number of bees for the *P*-*e* individuals.

```
Initialize PS scout bees;
While (Termination criterion is not met)
    % Sort scout bees
    Sort the scout bees based on the objective values, where the best solution is in the first position;
    For any individual $p$ in $[1, \cdots, e]$    % Update the best employed bees
        Achieve a set of nep neighbor solutions of individual $p$ and select the best one ($p'$);
        $p \leftarrow p'$ when $p'$ obtains a better fitness value;
    Endfor
    For any individual $p$ in $[e + 1, \cdots, P]$    % Update the remaining employed bees
        Achieve a set of nsp neighbor solutions of individual $p$ and select the best one ($p'$);
        $p \leftarrow p'$ when $p'$ obtains a better fitness value;
    Endfor
    For any individual $p$ in $[P + 1, \cdots, PS]$    % Update scouts
        Replace the solution $p$ with a new randomly generated individual;
    Endfor
Endwhile
```

Output: The best individual

4.6 Procedure of the TBA

This section presents the running mechanism of TBA, which is an improved version of OBA to accelerate the evolution process and the exploration capacity. The two-phase procedure of the TBA is illustrated in Algorithm 6, where a new parameter called *limit* is utilized. As seen, the proposed TBA makes modifications in sorting scout bees as well as updating the best employed bees, the remaining employed bees and scouts. It also abandons the solutions that have not been improved for *limit* times consecutively.

Algorithm 6: The main procedure of the TBA

% Parameter *limit* indicates that one individual should be abandoned when it has not been improved for *limit* times consecutively.

```
Initialize PS scout bees;
While (Termination criterion is not met)
    % Sort scout bees
    Set the objective values of the duplicated individuals as large positive numbers;
    Sort the scout bees based on the objective values, so that the best solution is in the first position;
    For any individual $p$ in $[1,\cdots,e]$    % Update the best employed bees
        For $k$:=1 to nep
            Achieve a neighbor solution $p'$ of individual $p$;
            $p \leftarrow p'$ when $p'$ obtains a better or the same fitness value;
        Endfor
    Endfor
    For any individual $p$ in $[e+1,\cdots,P]$    % Update the remaining employed bees
        For $k$:=1 to nsp
            Achieve a neighbor solution $p'$ of individual $p$;
            $p \leftarrow p'$ when $p'$ obtains a better or the same fitness value;
        Endfor
    Endfor
    For any individual $p$ in $[P+1,\cdots,PS]$    % Update scouts
        Select one individual $s$ from the best $P$ individuals randomly;
        Achieve a set of neighbor solutions of individual $s$ after applying neighbor operators for several times;
        Select the best neighbor solution $s'$ (different from $s$) to replace the solution $p$;
    Endfor
    For any individual $p$ in $[1,\cdots,e]$ which has not been improved for limit times consecutively
        Achieve a set of neighbor solutions of individual $p$ after applying neighbor operators for several times;
        Select the best neighbor solution $p'$ (different from $p$) to replace solution $p$;
    Endfor
Endwhile
```

Output: The best individual

In sorting scout bees, the TBA first sets the objective values of the duplicated individuals as large positive numbers. This modification ensures that the duplicated individuals are abandoned and replaced to help the algorithm escape from being trapped into local optima. In updating the employed bees, incumbent individual is replaced with a new neighbor solution when the neighbor solution obtains the same or better objective values. Here, a new acceptance criterion is utilized as many solutions may have the same objective values. The modification aims at accelerating the evolution process by avoiding the ineffective search around a poor-quality solution.

In updating the scouts, the abandoned individuals are replaced with the best one among a set of neighbor solutions applying neighbor operators for several times. This modification helps to bring new high-quality solutions to achieve a high-quality and diverse swarm. Especially, to avoid being trapped into local optima, this algorithm also abandons the solutions that have not been improved for *limit* times consecutively. In short, all these modifications are made with the purpose of achieving a proper balance between exploitation and exploration.

5 Computational study

This section tests the performances of the proposed methodologies. Two sets of instances are solved: the first set is taken from the published papers studying DLBP and the second set is modified from ALBP by Kalayci et al. (2016).

The first set contains P8 based on a PC disassembly process with 8 parts (Kalayci and Gupta 2014), P10 with 10 tasks (Kalayci and Gupta 2013b), P10-OR with 10 tasks, P25 with 25 subassemblies (Kalayci and Gupta 2013b), and P47 based on a laptop disassembly process with 47 parts (Kalayci et al. 2015). As there are three sets of removal times in Kalayci et al. (2015) for fuzzy DLBP, this section tests three cases with different task operation times: P47A, P47B and P47C. In total, the first set contains eight instances and the detailed data of these instances are available upon request. Notice that there are both AND precedence relationships and OR precedence relationships in P10-OR, and hence this instance is highlighted by adding "-OR".

The second set contains a total number of 47 instances, where the largest-size instance has 148 tasks and there is only AND precedence relationship in these 47 instances.

The proposed TABC and TBA are compared with 13 other algorithms: tabu search algorithm (TS), iterated local search algorithm (ILS) (Li et al. 2019b), genetic algorithm (GA), particle swarm optimization algorithm (PSO), discrete particle swarm optimization algorithm (DPSO), teaching–learning-based optimization algorithm (TLBO), original cuckoo search algorithm (OCS), discrete cuckoo search algorithm (DCS) (Li et al. 2019a), improved migrating birds optimization (IMBO) (Li et al. 2019a), original artificial bee colony algorithm (OABC), two improved artificial bee colony algorithms (ABC1 and ABC2) (Li et al. 2019a) and original bee algorithm (OBA). All the algorithms are re-implemented utilizing the same programming language (C++) and detailed procedures of the implemented algorithms are available upon request.

All the algorithms solve the instances in 10 repetitions to have enough data to conduct the statistical analysis. The algorithms terminate when the elapsed computation time reaches to 100 s (s) for the small-size instances with less than 70 tasks or 600 s for the large-size instances with larger than or equal to 70 tasks. All the algorithms are implemented utilizing the C++ programing language of the Microsoft Visual Studio 2015. The MILP model is solved utilizing the CPLEX solver of the General Algebraic Modeling System 23.0 and it terminates when the optimal solution is obtained and verified or the computation time reaches to 3600 s. All the experiments are conducted on a set of virtual computers with the same setting (one virtual processor and 2 GB RAM memory). These virtual computers are generated on a tower type of server, which has two Intel Xeon E5-2680 v2 processors and 64 GB RAM memory.

Section 5.1 presents two case studies to highlight the difference between DLBP and UDLBP. Subsequently, Sect. 5.2 evaluates the two-phase algorithms and Sect. 5.3 presents the results of the comparative study.

5.1 Case studies

This section presents two case studies to solve P10-OR and P25; respectively. Here, P10-OR is solved first. The precedence relationships diagram and knowledge database of P10-OR have already been provided in Fig. 2 and Table 2 (in previous sections), respectively. To evaluate the benefits of the U-shaped layout, both DLBP and UDLBP are solved and the results of these two layouts are compared. Table 3 presents the corresponding objective functions, where *Best*, *Avg* and *S.D.* are the best, average and the standard deviation of the corresponding objective values. Notice that TABC and TBA show the same performance for this small-size instance.

From Table 3, it is clear that CPLEX can solve the problem optimally (with 5 workstations). The proposed TABC and TBA also obtain the optimal station number in each repletion of 10 times. As the values of *S.D.* of all objectives are 0.0, it might be concluded that the proposed TABC and TBA are robust in solving this instance. It is also observed that the U-shaped line has the same values of f_1 and f_2 with the straight line whereas U-shaped line has smaller values of f_3 and f_4. That means it is possible to remove hazardous tasks and tasks with larger demands faster in the U-shaped disassembly line. The reason lies behind is that DLBP might be regarded as the special situation of UDLBP, and U-shaped disassembly line has much larger solution space and thus higher flexibility.

The second case is P25 (based on a Samsung SCH-3500 cell phone) with a cycle time of 18 time-units (Kalayci and Gupta 2013b). The precedence relationships diagram is given in Fig. 7 and the corresponding database is presented in Table 4. It is clear that there are 25 tasks in total and only AND precedence relationships exist in the precedence diagram.

Table 3 Results for P10-OR by CPLEX and TABC/TBA algorithms

Layout	Method	Evaluation criteria	f_1	CPU	f_2	f_3	f_4
UDLBP	CPLEX	–	5	0.26	–	–	–
DLBP	TABC/TBA	Best	5	–	149	5	6090
		Avg	5	–	149	5	6090
		S.D	0.0	–	0.0	0.0	0.0
UDLBP	TABC/TBA	Best	5	–	149	3	5250
		Avg	5	–	149	3	5250
		S.D	0.0	–	0.0	0.0	0.0

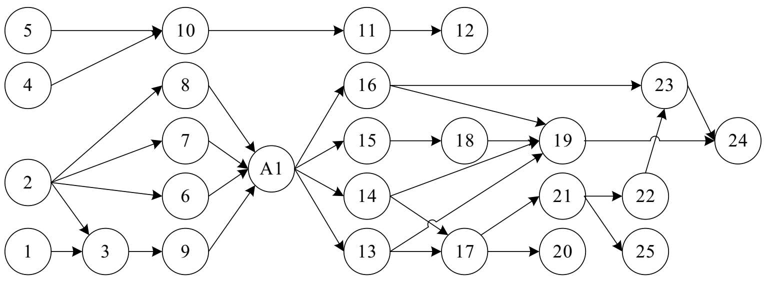

Fig. 7 The precedence diagram of P25 Kalayci and Gupta (2013b)

Table 4 Database for P25 Kalayci and Gupta (2013b)

Task	Part name	Part removal time	Hazardous	Demand
1	Antenna	3	Yes	4
2	Battery	2	Yes	7
3	Antenna guide	3	No	1
4	Bolt (type 1) A	10	No	1
5	Bolt (type1) B	10	No	1
6	Bolt (type2) 1	15	No	1
7	Bolt (type2) 2	15	No	1
8	Bolt (type2) 3	15	No	1
9	Bolt (type2) 4	15	No	1
10	Clip	2	No	2
11	Rubber seal	2	No	1
12	Speaker	2	Yes	4
13	White cable	2	No	1
14	Red/blue cable	2	No	1
15	Orange cable	2	No	1
16	Metal top	2	No	1
17	Front cover	2	No	2
18	Back cover	3	No	2
19	Circuit board	18	Yes	8
20	Plastic screen	5	No	1
21	Keyboard	1	No	4
22	LCD	5	No	6
23	Sub-keyboard	15	Yes	7
24	Internal IC board	2	No	1
25	Microphone	2	Yes	4

Table 5 presents the results by CPLEX and TABC/TBA algorithm. Here, TABC and TBA obtain the same performance for this small-size instance again. From Table 5, it is observed that CPLEX obtains the optimal station number within 1.1 s.

Table 5 Results of P25 by CPLEX and TABC/TBA algorithm

Layout	Method	Evaluation criteria	f_1	CPU	f_2	f_3	f_4
UDLBP	CPLEX	–	9	1.11	–	–	–
DLBP	TABC/TBA	Best	9	–	9	76	825
		Avg	9.00	–	9.00	76.00	825.00
		S.D	0.00	–	0.00	0.00	0.00
UDLBP	TABC/TBA	Best	9	–	**7**	**71**	873
		Avg	9.00	–	**7.00**	**71.00**	873.00
		S.D	0.00	–	0.00	0.00	0.00

*Best in bold

The proposed TABC and TBA obtain the optimal station number in each repetition. U-shaped layout provides a better line balance with a smaller value of f_2. As the objectives are considered hierarchically, it is sufficient to state that the U-shaped disassembly line outperforms the straight disassembly line and has a better line balance and idle time distribution. Still, it is observed that U-shaped line also obtains a smaller value of f_3, indicating that the U-shaped disassembly line has a strong ability of removing the hazardous tasks at earlier stations.

5.2 Evaluation of the two-phase algorithms

This section evaluates the performances of the two-phase algorithms. The proposed TABC and TBA are compared with methods without two-phase evaluation: ABC and BA with hierarchical objective functions to handle multiple objectives. Table 6 presents the average station numbers by ABC, BA, TABC and TBA. As seen in the table, TABC and TBA obtain the same or better results than ABC and BA for all the instances. Specifically, for the largest-size problem Barthol2, TABC and TBA outperform ABC and BA by a significant margin for all the four instances. In summary, this comparative study demonstrates that the two-phase evaluation enhances the algorithms' performance in minimizing the number of workstations. As a consequence, the utilization of two-phase evaluation mechanism is reasonable.

5.3 Comparative study

This section tests the performances of the proposed TABC and TBA by comparing with 13 other re-implemented algorithms. Due to page limits, this section mainly provides the results in terms of the station number (f_1) as it has the highest priority in the hierarchy. Note that the detailed results in terms of other objectives are available upon request. Table 7 illustrates the average station number (f_1) by implemented algorithms. Here, only the results by 10 algorithms are provided for space reasons. As you can see, the algorithms obtain similar performance when solving the small-size instances. But they have different performances when solving the very large-size instances. For example, TABC is the best performer for the Barthol2 with

Table 6 Average station number by ABC, BA, TABC and TBA

Instance	N	CT	ABC	BA	TABC	TBA
P10-OR	10	40	5	5	5	5
P8	8	40	4	4	4	4
P10	10	40	5	5	5	5
P25	25	18	9	9	9	9
P47A	47	105	7	7	7	7
P47B	47	106	9	9	9	9
P47C	47	120	9	9	9	9
Mertens	7	7	5	5	5	5
Bowman	8	20	4	4	4	4
Jaeschke	9	7	7	7	7	7
Jackson	11	10	5	5	5	5
Mansoor	11	94	2	2	2	2
Mitchell	21	15	8	8	8	8
Roszieg	25	16	8	8	8	8
Heskiaoff	28	216	5	5	5	5
Buxey	29	30	11.8	11.8	**11**	**11**
Lutz1	32	2357	7	7	7	7
Gunther	35	41	13	13	**12**	**12**
Kilbridge	45	62	9.9	9.7	**9**	**9**
Hahn	53	2806	6	6	**5.5**	**5.8**
Tonge	70	168	22.2	22.1	**22**	**21.9**
Tonge	70	170	22	22.1	**21.7**	**21.7**
Tonge	70	173	22	22	21	21
Tonge	70	179	21	21	20	20
Tonge	70	182	20.7	20.7	**20**	**20**
Wee-Mag	75	46	35.7	35.4	**34**	**34**
Wee-Mag	75	47	33.4	33.6	**33**	**33**
Wee-Mag	75	49	32.2	32.3	**32**	**32**
Wee-Mag	75	50	32	32	32	32
Wee-Mag	75	52	31.8	31.9	**31**	**31**
Arcus1	83	3985	20.1	20.1	**20**	**20**
Arcus1	83	5048	16	16	16	16
Arcus1	83	5853	14	14	**13**	**13**
Arcus1	83	6842	12	12	12	12
Arcus1	83	7571	11	11	11	11
Arcus1	83	8412	10	10	10	10
Arcus1	83	8898	9	9	9	9
Arcus1	83	10,816	8	8	**8**	**7.8**
Lutz2	89	15	33.6	33.8	**33**	**33**
Lutz3	89	150	12	12	**11**	**11**
Mukherjee	94	201	22	22	**21**	**21**
Mukherjee	94	301	15	15	**14**	**14**
Arcus2	111	5755	27.8	27.8	**27**	**27**

Table 6 (continued)

Instance	N	CT	ABC	BA	TABC	TBA
Arcus2	111	7520	21.1	21.1	**21**	**21**
Arcus2	111	8847	18	18	18	18
Arcus2	111	10,027	16	16	16	16
Arcus2	111	10,743	15	15	15	15
Arcus2	111	11,378	14	14	14	14
Arcus2	111	11,570	14	14	14	14
Arcus2	111	17,067	9	9	9	9
Barthol2	148	85	52	52	**51**	**51**
Barthol2	148	89	50	50	**48.7**	**48.5**
Barthol2	148	91	48.6	49	**47.1**	**47.5**
Barthol2	148	95	46.9	46.5	**45.7**	**45.3**

*Improvements in bold

a cycle time of 91, TBA is the best performer for the Barthol2 with a cycle time of 89 and 95. When comparing OABC and TABC, it is observed that TABC obtains the same or better results for all the tested instances, indicating that the proposed improvements enhance the TABC by a significant margin. Meanwhile, TBA also obtains the same results with or better results than OBA for all the tested instances, denoting that the proposed improvements significantly enhance the performance of TBA.

To have a better observation of the results, Table 8 presents the summary of the results by 13 algorithms in comparison with ILS. In this table, *#Better-than-ILS* reports the number of instances for which one algorithm outperforms ILS over 54 test cases. *#Equal-to-ILS* column reports the number of instances for which one algorithm has the same performance with ILS, and *#Worse-than-ILS* reports the number of instances for which ILS outperforms the corresponding algorithm. Regarding the number of stations (f_1), ILS outperforms TS, GA, PSO, DPSO, TLBO, OCS, DCS, IMBO, OABC, ABC1, ABC2 and OBA; but ILS is outperformed by TABC and TBA. Regarding the objectives f_1,f_2, f_3 and f_4, ILS outperforms TS, GA, PSO, DPSO, TLBO, OCS, DCS, IMBO, OABC, ABC1, ABC2 and OBA; but ILS is outperformed by TABC and TBA. As a result, it can be concluded that TABC and TBA are the two best performers and outperform their original versions.

The Friedman rank-based analysis is conducted to evaluate the algorithms statistically. The algorithm types are considered as the controlled factor and the station number (f_1) is selected as the response variable. The statistical analysis demonstrates that there is a statistically significant difference between the performances of the tested algorithms. The mean ranks of the TS, ILS, GA, PSO, DPSO, TLBO, OCS, DCS, IMBO, OABC, ABC1, ABC2, OBA, TABC and TBA are 8.019, 7.389, 9.667, 7.870, 9.269, 7.491, 8.380, 7.556, 7.843, 8.630, 7.556, 7.611, 8.500, 7.222 and 7.000, respectively. Among all the algorithms, TBA is the best performer, TABC is the second-best performer and ILS is the third-best performer. Figure 8 illustrates the ranks in term of station number (f_1). TABC and TBA show slightly better performance than TS, GA, PSO, DPSO, TLBO, OCS, DCS, IMBO, OABC, ABC1,

Table 7 Average station number by implemented algorithms

Instance	CT	TS	ILS	PSO	IMBO	OABC	ABC1	ABC2	OBA	TABC	TBA
P10-OR	40	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
P8	40	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
P10	40	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
P25	18	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0
P47A	105	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0
P47B	106	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0
P47C	120	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0
Mertens	7	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
Bowman	20	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
Jaeschke	7	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0
Jackson	10	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
Mansoor	94	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
Mitchell	15	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0
Roszieg	16	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0
Heskiaoff	216	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
Buxey	30	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0
Lutz1	2357	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.0
Gunther	41	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0
Kilbridge	62	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0
Hahn	2806	5.7	5.6	5.9	5.8	5.1	5.9	5.3	6.0	5.5	5.8
Tonge	168	22.0	22.0	21.9	22.0	22.0	22.0	22.0	22.0	22.0	21.9
Tonge	170	22.0	21.8	21.0	21.7	22.0	21.5	21.9	21.9	21.7	21.7
Tonge	173	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0
Tonge	179	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
Tonge	182	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
Wee-Mag	46	34.0	34.0	34.0	34.0	34.4	34.0	34.0	34.0	34.0	34.0
Wee-Mag	47	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0
Wee-Mag	49	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0
Wee-Mag	50	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0
Wee-Mag	52	31.0	31.0	31.0	31.0	31.0	31.0	31.0	31.0	31.0	31.0
Arcus1	3985	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0
Arcus1	5048	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0
Arcus1	5853	13.0	13.0	13.3	13.1	13.9	13.0	13.1	13.8	13.0	13.0
Arcus1	6842	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0
Arcus1	7571	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0
Arcus1	8412	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
Arcus1	8898	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0
Arcus1	10,816	8.0	7.7	8.0	8.0	8.0	8.0	8.0	8.0	8.0	7.8
Lutz2	15	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0	33.0
Lutz3	150	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0	11.0
Mukherjee	201	21.9	21.0	21.8	21.5	22.0	21.4	21.4	22.0	21.0	21.0
Mukherjee	301	14.0	14.0	14.3	14.1	15.0	14.0	14.0	14.9	14.0	14.0

Table 7 (continued)

Instance	CT	TS	ILS	PSO	IMBO	OABC	ABC1	ABC2	OBA	TABC	TBA
Arcus2	5755	27.0	27.0	27.0	27.0	27.0	27.0	27.0	27.0	27.0	27.0
Arcus2	7520	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0
Arcus2	8847	18.0	18.0	18.0	18.0	18.0	18.0	18.0	18.0	18.0	18.0
Arcus2	10,027	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0	16.0
Arcus2	10,743	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0
Arcus2	11,378	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0
Arcus2	11,570	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0
Arcus2	17,067	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0
Barthol2	85	51.0	51.0	51.0	51.0	51.6	51.0	51.0	51.0	51.0	51.0
Barthol2	89	49.0	48.8	49.0	48.7	49.0	49.0	49.0	49.0	48.7	48.5
Barthol2	91	48.0	47.8	47.9	48.0	48.0	47.7	47.7	48.0	47.1	47.5
Barthol2	95	46.0	46.0	45.8	45.8	46.0	45.5	45.7	46.0	45.7	45.3

Table 8 Summary of the results by algorithms in comparison with ILS

Algorithms	f_1			f_1,f_2, f_3 and f_4		
	#Better-than-ILS	#Equal-to-ILS	#Worse-than-ILS	#Better-than-ILS	#Equal-to-ILS	#Worse-than-ILS
TS	0	48	6	8	8	38
GA	0	38	16	7	8	39
PSO	3	44	7	6	8	40
DPSO	0	41	13	12	9	33
TLBO	4	43	7	10	8	36
OCS	1	45	8	7	9	38
DCS	2	48	4	11	8	35
IMBO	3	45	6	5	8	41
OABC	1	44	9	16	9	29
ABC1	3	47	4	7	8	39
ABC2	3	46	5	10	8	36
OBA	0	46	8	16	9	29
TABC	**5**	**48**	**1**	**25**	**10**	**19**
TBA	**5**	**47**	**2**	**23**	**9**	**22**

Best in bold

ABC2 and OBA. Notice that, most algorithms can obtain the optimal station numbers for the "easy" instances and this situation could prevent observing the superior performances of the proposed TABC and TBA algorithms. In summary, this comparative study demonstrates that TABC and TBA outperform their original versions and achieve a competing performance in comparison with other 13 algorithms.

In summary, the case studies demonstrate that the proposed MILP model obtains the optimal station numbers for all the tested small-size instances. The U-shaped

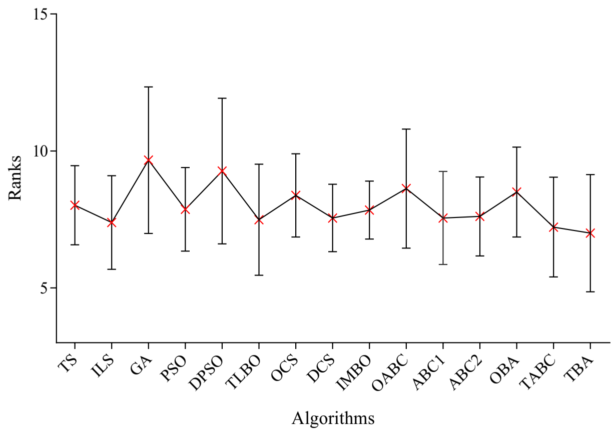

Fig. 8 Means plot of the average ranks and 95% confidence intervals of the algorithms

disassembly line obtains better fitness values than the straight disassembly line due to higher flexibility, and the U-shaped layout has a better line balance or idle time distribution along with the strong ability of removing the hazardous tasks and tasks with larger demand at first. In addition, the proposed TABC and TBA are capable of obtaining the optimal solution regarding the number of stations verified by the MILP model, and these algorithms are robust in solving small-size instances. The computational and comparative studies show that the improvements enhance the search capability of TABC and TBA by a significant margin. TABC and TBA obtain competing performance in comparison with 13 other algorithms, demonstrating that the TABC and TBA are quite effective and efficient for multi-objective UDLBP.

6 Conclusions and future research

U-shaped disassembly lines have higher flexibility than straight disassembly lines and have drawn increasing attention from industry. This study addressed the U-shaped disassembly line balancing problem (UDLBP) with AND/OR precedence relationships. A mixed-integer programming model is developed to formulate the AND/OR precedence relationships with the objective of minimizing the number of workstations. This model is then extended to a mixed-integer nonlinear programming model to optimize four objectives. To solve the multi-objective UDLBP, a two-phase artificial bee colony algorithm (TABC) and a two-phase bee algorithm (TBA) are developed and improved. The proposed TABC utilizes a modified employed bee phase with a new acceptance criterion, modified onlooker phase to replace the poor-quality solutions, modified scout phase to obtain a high-quality swarm with certain diversities and optionally local search to emphasize exploitation. Meanwhile, the proposed TBA utilizes a new employed bee phase to speed up the search process and a new scout phase to achieve high-quality and diverse swarm. These help the algorithm to escape from local optima. All the modifications help the proposed TABC and TBA to obtain a proper balance between exploration and exploitation.

Case studies show that the proposed MILP model utilizing CPLEX solver obtains the optimal solutions for the tested small-size instances regarding the station number, and U-shaped disassembly lines achieve better fitness values than the straight disassembly lines due to higher flexibility. Computational study demonstrates that the utilization of a two-phase approach is reasonable and the improvements enhance the performances of TABC and TBA by a significant margin. Meanwhile, the proposed algorithms obtain a promising performance in comparison to 13 other algorithms. As a consequence, the proposed TABC and TBA are capable of solving the multi-objective UDLBP efficiently and effectively.

Through a managerial point of view, it is reasonable to prefer the U-shaped layout rather than a straight line as long as the physical conditions allow. So that the line managers could increase their throughput rate using the same workforce. It is also possible to minimize the workforce requirement and so the costs thanks to the efficiency that the U-shaped layout provides. The methods proposed in this research can also be of interest for practitioners to achieve efficient solutions regardless of the line layout chosen.

One of the main limitations of this work could be the lack of real data, which deserves a comprehensive effort considering commercial confidentiality. Furthermore, it would be beneficiary to solve more instances with OR precedence relationships.

Future research stems from applying the developed algorithms to tackle the extensions of the UDLBP by considering more realistic constraints, including stochastic UDLBP, partial UDLBP and many others. As there is no exact method to solve the UDLBP, it might be interesting to develop exact methods, such as branch and bound algorithm, to solve the UDLBP optimally. It is also suggested to develop Pareto algorithms to obtain a set of non-dominated solutions in solving the multi-objective UDLBP.

Appendix

Appendix A: MILP model retrieved from Wang et al. (2020)

This model is retrieved from Wang et al. (2020). The notations are as follows.

Indices	
i, j, h	Task/part index, $i, j, h \in \{1, 2, \dots, N\}$, where N is the number of real and dummy tasks/parts
p	Station index, $p \in \{1, 2, \dots, M\}$, where M is the maximum number of stations allowed to be opened
CT	Cycle time
t_i	Operation time of performing task i or part removal time of part i
ANDP(i)	Set of AND predecessors of task i
Decision variables	

A_{ip}	1, if task/part i is allocated to the entrance side of station p; 0, otherwise
B_{ip}	1, if task/part i is allocated to the exit side of station p; 0, otherwise
z_p	1, if station p is opened; 0, otherwise

This model, referred to as Model 1, is formulated utilizing expressions (18–23). This model is a MILP model and the objective function given in Eq. (18) optimizes the number of stations. Constraint (19) ensures that a task must be allocated to a station on the entrance side or the exit side. Constraint (20) tackles the cycle time constraint, indicating that the total operation time of tasks in the entrance and exit sides of one station is less than or equal to the cycle time given. Constraint (21) and constraint (22) together deal with AND precedence relationships. Constraint (21) indicates that predecessor i of task j should be allocated to the former or the same station when task i and task j are allocated to the entrance side. Constraint (22) indicates that predecessor i of task j should be allocated to the latter or the same station when task i and task j are allocated to the exit side. Constraint (23) restricts the decision variables to be binary.

$$\mathrm{Min}\, f_1 = \sum_{p=1}^{M} z_p \tag{18}$$

$$\sum_{p}^{M} \left(A_{\mathrm{ip}} + B_{\mathrm{ip}}\right) = 1 \quad \forall i \tag{19}$$

$$\sum_{i=1}^{N} t_i \cdot \left(A_{\mathrm{ip}} + B_{\mathrm{ip}}\right) \le z_p \cdot CT \quad \forall p \tag{20}$$

$$\sum_{p}^{M} (M - p + 1) \cdot \left(A_{\mathrm{ip}} - A_{\mathrm{jp}}\right) \ge 0 \quad \forall j, i \in \mathrm{ANDP}(j) \tag{21}$$

$$\sum_{p}^{M} (M - p + 1) \cdot \left(B_{\mathrm{jp}} - B_{\mathrm{ip}}\right) \ge 0 \quad \forall j, i \in \mathrm{ANDP}(j) \tag{22}$$

$$A_{\mathrm{ip}}, B_{\mathrm{ip}}, y_p \in \{0, 1\} \quad \forall i, p \tag{23}$$

Acknowledgements This project is partially supported by National Natural Science Foundation of China Under Grants 62173260 and 61803287. The authors are grateful for the valuable comments by the anonymous referees to improve this paper.

References

Agrawal S, Tiwari MK (2008) A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. Int J Prod Res 46(6):1405–1429

Altekin FT, Akkan C (2012) Task-failure-driven rebalancing of disassembly lines. Int J Prod Res 50(18):4955–4976

Altekin FT, Kandiller L, Ozdemirel NE (2008) Profit-oriented disassembly-line balancing. Int J Prod Res 46(10):2675–2693

Çil ZA, Mete S, Serin F (2020) Robotic disassembly line balancing problem: a mathematical model and ant colony optimization approach. Appl Math Model 86:335–348

Deniz N, Ozcelik F (2019) An extended review on disassembly line balancing with bibliometric and social network and future study realization analysis. J Clean Prod 225:697–715

Ding L-P, Feng Y-X, Tan J-R, Gao Y-C (2010) A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. Int J Adv Manuf Technol 48(5):761–771

Fang Y, Liu Q, Li M, Laili Y, Pham DT (2019) Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations. Eur J Oper Res 276(1):160–174

Fang Y, Ming H, Li M, Liu Q, Pham DT (2020) Multi-objective evolutionary simulated annealing optimisation for mixed-model multi-robotic disassembly line balancing with interval processing time. Int J Prod Res 58(3):846–862

Gao KZ, He ZM, Huang Y, Duan PY, Suganthan PN (2020) A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. Swarm Evolut Comput 57:100719

Güngör A, Gupta SM (1999) Disassembly line balancing. In: Proceedings of the annual meeting of the northeast decision sciences Institute. Newport, RI, pp. 193–195

Gungor A, Gupta SM (2001) A solution approach to the disassembly line balancing problem in the presence of task failures. Int J Prod Res 39(7):1427–1467

Güngör A, Gupta SM (2002) Disassembly line in product recovery. Int J Prod Res 40(11):2569–2589

Kalayci CB, Gupta SM (2013a) Ant colony optimization for sequence-dependent disassembly line balancing problem. J Manuf Technol Manag 24(3):413–427

Kalayci CB, Gupta SM (2013b) Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. Exp Syst Appl 40(18):7231–7241

Kalayci CB, Gupta SM (2013c) A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. Int J Adv Manuf Technol 69(1):197–209

Kalayci CB, Gupta SM (2014) A tabu search algorithm for balancing a sequence-dependent disassembly line. Prod Plann Control 25(2):149–160

Kalayci CB, Hancilar A, Gungor A, Gupta SM (2015) Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm. J Manuf Syst 37:672–682

Kalayci CB, Polat O, Gupta SM (2016) A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. Ann Oper Res 242(2):321–354

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 Trans 41(10):866–881

Kucukkoc I (2020) Balancing of two-sided disassembly lines: problem definition, MILP model and genetic algorithm approach. Comput Oper Res 124:105064. https://doi.org/10.1016/j.cor.2020.105064

Kucukkoc I, Buyukozkan K, Satoglu SI, Zhang DZ (2019) A mathematical model and artificial bee colony algorithm for the lexicographic bottleneck mixed model assembly line balancing problem. J Intell Manuf 30(8):2913–2925. https://doi.org/10.1007/s10845-015-1150-5

Kucukkoc I, Li Z, Li Y (2020) Type-E disassembly line balancing problem with multi-manned workstations. Optim Eng 21:611–630

Li J, Chen X, Zhu Z, Yang C, Chu C (2019c) A branch, bound, and remember algorithm for the simple disassembly line balancing problem. Comput Oper Res 105:47–57

Li Z, Janardhanan MN, Tang Q, Ponnambalam SG (2019a) Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. Swarm Evolut Comput 50:100567

Li Z, Kucukkoc I, Zhang Z (2019b) Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem. Comput Ind Eng 137:106056

Li Z, Çil ZA, Mete S, Kucukkoc I (2020) A fast branch, bound and remember algorithm for disassembly line balancing problem. Int J Prod Res 58(11):3220–3234

Liu J, Wang S (2017) Balancing disassembly line in product recovery to promote the coordinated development of economy and environment. Sustainability 9(2):309

Liu J, Zhou Z, Pham DT, Xu W, Yan J, Liu A, Ji C, Liu Q (2018) An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing. Int J Adv Manuf Tech 97(9–12):3937–3962

Liu M, Liu X, Chu F, Zheng F, Chu C (2019) Robust disassembly line balancing with ambiguous task processing times. Int J Prod Res. https://doi.org/10.1080/00207543.2019.1659520

Liu J, Zhou Z, Pham DT, Xu W, Ji C, Liu Q (2020) Collaborative optimization of robotic disassembly sequence planning and robotic disassembly line balancing problem using improved discrete Bees algorithm in remanufacturing. Robot Comput Integr Manuf 61:101829

McGovern SM, Gupta SM (2003) 2-opt heuristic for the disassembly line balancing problem. In: Proceedings of the SPIE international conference on environmentally conscious manufacturing III. Providence, RI, pp. 71–84

McGovern SM, Gupta SM (2006) Ant colony optimization for disassembly sequencing with multiple objectives. Int J Adv Manuf Technol 30(5):481–496

McGovern SM, Gupta SM (2007a) A balancing method and genetic algorithm for disassembly line balancing. Eur J Oper Res 179(3):692–708

McGovern SM, Gupta SM (2007b) Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. Int J Prod Res 45(18–19):4485–4511

Mete S, Çil ZA, Ağpak K, Özceylan E, Dolgui A (2016) A solution approach based on beam search algorithm for disassembly line balancing problem. J Manuf Syst 41:188–200

Mete S, Çil ZA, Özceylan E, Ağpak K, Battaïa O (2018) An optimisation support for the design of hybrid production lines including assembly and disassembly tasks. Int J Prod Res 56(24):7375–7389

Özceylan E, Kalayci CB, Güngör A, Gupta SM (2018) Disassembly line balancing problem: a review of the state of the art and future directions. Int J Prod Res 57(15–16):4805–4827

Paksoy T, Güngör A, Özceylan E, Hancilar A (2013) Mixed model disassembly line balancing problem with fuzzy goals. Int J Prod Res 51(20):6082–6096

Ren Y, Yu D, Zhang C, Tian G, Meng L, Zhou X (2017) An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. Int J Prod Res 55(24):7302–7316

Ren Y, Zhang C, Zhao F, Tian G, Lin W, Meng L, Li H (2018a) Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-Optimal algorithm. J Clean Prod 174:1475–1486

Ren Y, Zhang C, Zhao F, Triebe MJ, Meng L (2018) An MCDM-based multiobjective general variable neighborhood search approach for disassembly line balancing problem. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/TSMC.2018.2862827

Wang K, Li X, Gao L (2019a) Modeling and optimization of multi-objective partial disassembly line balancing problem considering hazard and profit. J Clean Prod 211:115–133

Wang K, Li X, Gao L, Garg A (2019b) Partial disassembly line balancing for energy consumption and profit under uncertainty. Robot Comput Integr Manuf 59:235–251

Wang S, Guo X, Liu J (2019c) An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times. Eng Optim 51(11):1920–1937

Wang K, Gao L, Li X (2020) A multi-objective algorithm for U-shaped disassembly line balancing with partial destructive mode. Neural Comput Appl 32(16):12715–12736

Xiao S, Wang Y, Yu H, Nie S (2017) An entropy-based adaptive hybrid particle swarm optimization for disassembly line balancing problems. Entropy 19(11):596

Yang Y, Yuan G, Zhuang Q, Tian G (2019) Multi-objective low-carbon disassembly line balancing for agricultural machinery using MDFOA and fuzzy AHP. J Clean Prod 233:1465–1474

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. Exp Syst Appl 86:165–176

Zhang L, Zhao X, Ke Q, Dong W, Zhong Y (2019) Disassembly line balancing optimization method for high efficiency and low carbon emission. Int J Precis Eng Manuf-Green Technol. https://doi.org/10.1007/s40684-019-00140-2

Zhu L, Zhang Z, Wang Y (2018) A pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation. Int J Prod Res 56(24):7354–7374

Authors and Affiliations

Zixiang Li[1,2] · **Ibrahim Kucukkoc**[3] · **Qiuhua Tang**[1,2] · **Zikai Zhang**[1,2]

Zixiang Li
zixiangliwust@gmail.com

Qiuhua Tang
tangqiuhua@wust.edu.cn

Zikai Zhang
zhangzikai0703@gmail.com

1 Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan, Hubei, China

2 Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, Hubei, China

3 Department of Industrial Engineering, Balikesir University, Cagis Campus, 10145 Balıkesir, Turkey