Memetic Computing (2021) 13:111–130
https://doi.org/10.1007/s12293-020-00319-0

REGULAR RESEARCH PAPER

Local search methods for type I mixed-model two-sided assembly line balancing problems

Zixiang Li[1,2] · Mukund Nilakantan Janardhanan[3] · Qiuhua Tang[1,2] · Peter Nielsen[4]

Received: 20 March 2019 / Accepted: 14 December 2020 / Published online: 5 January 2021

Abstract
Two-sided assembly lines are widely utilized to assemble large-sized products such as cars and trucks. Recently, these types of assembly lines have been applied to assemble different types of products due to a large variety of customer demands and strong market competition. This paper presents two simple local search methods, the iterated greedy algorithm and iterated local search algorithm, to deal with type I mixed-model two-sided assembly line balancing problems. These two algorithms utilize new precedence-based local search functions with referenced permutation and two neighborhood structures to emphasize intensification while preserving high search speed. Additionally, these local search methods are enhanced by utilizing the best decoding scheme amongst nine candidates and a new station-oriented evaluation to guide the search direction. New lower bound calculations are also presented to check the optimality of the achieved solutions. Eleven recent and high-performing metaheuristic algorithms are re-implemented to test the performance of the proposed algorithms. A comprehensive study on a set of benchmark problems demonstrates the advantages of the improvements and the superiority of the two proposed methods. Experimental results show that the proposed algorithms obtain 23 new upper bounds compared with two recently published algorithms, among which 19 cases are proven to be optimal for the first time.

Keywords Metaheuristics · Assembly line balancing · Two-sided assembly line · Mixed-model production · Local search

1 Introduction

Assembly lines are of great importance to several industries such as the automotive and consumer electronics industries [3]. These types of production systems are mainly designed for producing high-quality and standardized products. In this type of line, a set of tasks is allocated to and operated on a set of workstations. Each task has a deterministic and non-negative operation time, and each workstation is connected using a transportation system such as a conveyor belt. The total operation time of the tasks in a workstation is less than a limited time referred to as the cycle time. The assembly line balancing problem aims at determining the best allocation of tasks to the workstations with the optimization of one or several objective functions.

The simple assembly line balancing problem has been criticized for being too theoretical, as most real industrial settings are much more complex. Further, variants of this problem have attracted significant attention with the goal of reducing the gap between real-world application and research. Among the more complex real-world assembly line configurations, the two-sided variant is a good choice

✉ Mukund Nilakantan Janardhanan
mukund.janardhanan@leicester.ac.uk

Zixiang Li
zixiangliwust@gmail.com

Qiuhua Tang
tangqiuhua@wust.edu.cn

Peter Nielsen
peter@mp.aau.dk

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 Mechanics of Materials Research Group, Department of Engineering, University of Leicester, Leicester, UK

4 Department of Materials and Production, Aalborg University, Aalborg, Denmark

for manufacturing large products such as trucks and cars. In a two-sided assembly line, unlike the traditional one-sided assembly line, a set of tasks are allocated to a set of mated-stations, and each mated-station is composed of two opposite-facing workstations. There are three types of tasks on this line: L-type tasks that must be allocated to the left side, R-type tasks that must be allocated to the right side, and E-type tasks that can be allocated to either side. This type of assembly line has several advantages when compared with traditional one-sided assembly lines: shorter length, reduced setup time, and the ability for increased sharing of tools [2]. The two-sided assembly line balancing problems, referred to as TALBP, are highly complex due to the utilization of two sides of mated-stations and the existence of three types of tasks.

Mixed-model assembly lines are capable of assembling more than one product with similar characteristics simultaneously. Assembly lines of this type can be found in the automotive industry, furniture factories, and so forth. The major goal of the mixed-assembly line balancing is to handle diverse products while achieving high line efficiency. From the literature, it can be seen that both two-sided assembly line balancing and mixed-model assembly lines have been separately and thoroughly studied. However, studies of mixed-models in a two-sided assembly line balancing problem (MTALBP) have been limited, even though these types of assembly lines are widely used in the automotive industry. Hence, this paper focuses mainly on the type I mixed-model two-sided assembly line balancing problem (MTALBP-I) with the objective of minimizing the number of mated-stations and the number of workstations.

As the well-known simple one-sided assembly line balancing is already NP-hard in the strong sense [3], the considered MTALBP-I is also NP-hard, and it is more complex due to the utilization of two sides and the consideration of a mixed-model. To the authors' best knowledge, most researchers have focused on the single-model TALBP and the research results reported with regard to the MTALBP are limited. Moreover, published research focuses on utilizing different heuristic methods to select the appropriate side for E-type tasks in their decoding procedures, and there has been no detailed comparison of such decoding schemes to identify the most effective one. Furthermore, the reported objectives might be ineffective in solving the MTALBP-I. In practice, many of the obtained solutions have the same number of workstations. Hence, the objective of minimizing the number of workstations cannot readily be reached as there exist a very large number of similar solutions. The weighted method of maximization of the line efficiency and minimizing the smoothness proposed by Özcan and Toklu [18] is also not effective in minimizing the workstation number since it might lose the ability to find the optimal solution, which is further discussed in Sect. 6.2.

To overcome the aforementioned drawbacks, this paper presents four major contributions. (1) Six new decoding schemes are developed for the first time to solve MTALBP-I and a comparative study of nine decoding procedures is also carried out to test their performance, where most of the reported decoding methods are covered. (2) An iterated greedy algorithm and iterated local search algorithm are extended and improved to solve the MTALBP for the first time. A new local search with two neighborhood structures is developed to enhance their performance. A two-stage evaluation procedure is developed to guide the search process, where the first stage with a station-oriented evaluation is applied to find the solutions with the fewest workstations, and the second stage attempts to find balanced solutions on the basis of the solution obtained in the first stage. (3) A set of 11 other algorithms is also implemented for the considered problem, among which nine algorithms are developed for the first time to solve the considered problem. A comprehensive study is conducted to evaluate all 13 algorithms. The computational results demonstrate the superiority of the proposed local search methods, and the proposed methods are able to obtain 23 new upper bounds. (4) A new lower bound calculation is presented to check the optimality of the obtained solutions. This method achieves 19 better lower bounds and confirms the optimality of 14 large-sized cases for the first time.

The remainder of this paper is organized as follows. Section 2 presents an overview of the reported literature. Section 3 presents the problem description and formulation. Section 4 introduces the decoding schemes, and Sect. 5 presents the two proposed algorithms with new local search strategies. Section 6 discusses the computational results of the new algorithms, and finally Sect. 7 presents the conclusions and provides future research directions.

2 Literature review

Assembly line balancing problems have been widely studied since the first study by Salveson in 1955 [3]. This section first introduces the heuristic techniques used to solve the TALBP and later discusses the recent research on the MTALBP.

TALBP was first introduced by Bartholdi [2], and a modified "first fit" heuristic was developed to solve this problem. Lee et al. [13] developed a group assignment procedure, and Özcan and Toklu [20] presented a new heuristic-based approach for balancing two-sided assembly lines with sequence-dependent setup times. As exact methods cannot solve the large-sized instance in an acceptable time, metaheuristics are widely employed. Kim et al. [8] developed a genetic algorithm to solve the TALBP with positional constraints, and the genetic algorithms were later proposed by Kim et al. [9] and Purnomo et al. [22] for solving the TALBP with a cycle time minimization criterion. Other algorithms such as the tabu search algorithm [19] and ant colony algorithm [4] were later developed to solve the TALBP. Khorasanian et al. [7] utilized a simulated annealing algorithm which thus far outperforms all other published methods. The higher performance of the method is partly attributed to the new decoding scheme and the new objective function. More recently, Li et al. [16] developed an iterated greedy algorithm to tackle the TALBP with a cycle time minimization criterion. A detailed summary of different algorithms which have been applied can be found in Li et al. [15].

Regarding the MTALBP, Simaria and Vilarinho [24] introduced the MTALBP, and proposed a mathematical programming model. Thereafter, Özcan and Toklu [18] developed a new mixed-integer programming model, and also developed a simulated annealing algorithm to solve the large-sized problems. Delice et al. [5] improved this algorithm, and also improved the decoding procedure reported in Özcan and Toklu [18] by allocating the E-type task to the side which can start the task earlier. This improved algorithm has been proven to outperform the results obtained using a simulated annealing algorithm, however the decoding procedures are not compared in detail. Aghajani et al. [1] considered the robotic mixed-model two-sided assembly line with robot setup times. Yuan et al. [27] improved Özcan and Toklu [18] decoding procedure by allocating E-type tasks to the side with a larger capacity. They also developed a hybrid honey bee mating optimization algorithm to maximize line efficiency and minimize weighted smoothness index. This new algorithm can find better results than the reported results, but decoding procedures have again not yet been compared. More recently, Li et al. [14] proposed a hybrid imperialist competitive algorithm to tackle a multi-objective MTALBP-I. Several other researchers have reported studies concerning the mixed-model parallel TALBP [12], MTALBP with underground workstations [10], MTALBP with setup time [26], and the mixed-model two-sided assembly line balancing and sequencing problem [11].

Based on the above literature review, it can be seen that the literature on the MTALBP is limited and the improvements on the results of the MTALBP are mainly related to the new objectives or decoding schemes. Moreover, no work has reported a detailed comparison of different applied decoding schemes. To overcome the aforementioned gaps in the literature, this paper presents several new decoding schemes, a new station-oriented evaluation, and two local search algorithms. A comprehensive study is also carried out to test the performance of these improvements.

3 Problem description and formulation

This section first introduces the problem description in Sect. 3.1 and then presents the mathematical formulation in Sect. 3.2.

3.1 Problem description

Mixed-model two-sided assembly lines assemble a set of similar product models according to the operators on a set of *Nm* mated-stations. Each model has its specific precedence relation and the operation times of different models can differ from one other. All the precedence graphs of the product models can be merged into a combined precedence diagram [18]. In this combined precedence diagram, the operation time of task *i* of model *m* can be equal to zero, denoting that model *m* does not process task *i*. The basic assumptions of the MTALBP, based on Özcan and Toklu [18], are summarized as follows.

(1) All products have similar characteristics, and the precedence relations of the models can be merged into a combined precedence diagram.
(2) Operators assemble the products in parallel on both sides of the mated-stations.
(3) The task times of models can differ from each other, but they are all deterministic.
(4) The tasks of each model must be finished within a predetermined cycle time in their workstations and different models have the same cycle time.
(5) The parallel workstation, travel time of operators, and the work-in-process inventory are not taken into account.

To highlight the features of the considered MTALBP, a typical example with 12 tasks and a cycle time of 6 is here provided. Table 1 illustrates the combined precedence relations, task directions, and task times. As can be seen, two types of products, model A and model B, are assembled and the operation time of a task completed on model B can differ from that of model A. For instance, there is no task 3 in model B whereas the operation time of task 3 in model A is 2. Figure 1 depicts the detailed task assignment in a mixed-model two-sided assembly line.

During the allocation of tasks to mated-stations there are three constraints which need to be satisfied. (1) Precedence constraint: the predecessor of one task must be allocated to the former mated-station or be operated before this task when the predecessor and this task are allocated to the same mated-station. As can be seen in Fig. 1, precedence constraint is satisfied as all the predecessors of

Table 1 Precedence relation, task directions, and task times

Task	Successors	Preferred direction	Task times	
			Model A	Model B
1	4	L	2	3
2	5	R	3	3
3	6	E	2	0
4	7	L	3	2
5	7, 8, 9	E	1	2
6	9	L	1	0
7	10	E	3	2
8	10	R	3	1
9	11	E	2	0
10	–	E	2	3
11	12	E	0	2
12	–	R	0	1

a given task is completed before operating the task. For instance, task 1 is the predecessor of task 4 and hence task 1 is completed before operating task 4. (2) Direction constraint: the L-type tasks must be allocated to the left side, R-type tasks must be allocated to the right side, and E-type tasks can be allocated to either side. For instance, task 1 is an L-type task and hence it is assigned to the left side. (3) Cycle time constraint: Tasks of each model on workstations must be finished within a pre-determined cycle time. Clearly, all the tasks of all models are completed within the given cycle time of 6 in this example, which can be achieved by considering the models separately in the decoding procedure. Notice that there are two types of idle times in the MTALBP: the idle time behind the workstation and the idle time in the middle of a workstation. The second idle time is the special idle time resulting from precedence constraint and is referred to as sequence-depended idle time. For instance, the idle time behind task 8 is the sequence-depended idle time for model B.

3.2 Mathematical formulation

The notations utilized in the model formulation are presented as follows.

Indices	
i, h, p	Task index
j, g	Mated-station index
m	Product model index
k, f	The side of the line; $k,f = \begin{cases} 1 & \text{if the side is left} \\ 2 & \text{if the side is right} \end{cases}$
(j, k)	The k side workstation of the mated-station j
Parameters	
I	Set of tasks in the combined precedence diagram; $I = \{1, 2, \ldots, i, \ldots, Nt\}$
J	Set of mated-stations; $J = \{1, 2, \ldots, j, \ldots, Nm\}$
M	Set of product models; $M = \{1, 2, \ldots, m, \ldots, Np\}$
AL	Set of tasks which should be operated at a left-side workstation; $AL \subseteq I$
AR	Set of tasks which should be performed at a right-side workstation; $AR \subseteq I$
AE	Set of tasks which can be performed on the left or right side of a mated-station; $AE \subseteq I$
P_0	Set of tasks that have no immediate predecessors
$P_a(i)$	Set of all predecessors of task i
$P(i)$	Set of immediate predecessors of task i
$S_a(i)$	Set of all successors of task i
$S(i)$	Set of immediate successors of task i
$K(i)$	Set of integers indicating the preferred operation direction of task i; $C(i) = \begin{cases} \{1\} & \text{if } i \in AL \\ \{2\} & \text{if } i \in AR \\ \{1,2\} & \text{if } i \in AE \end{cases}$

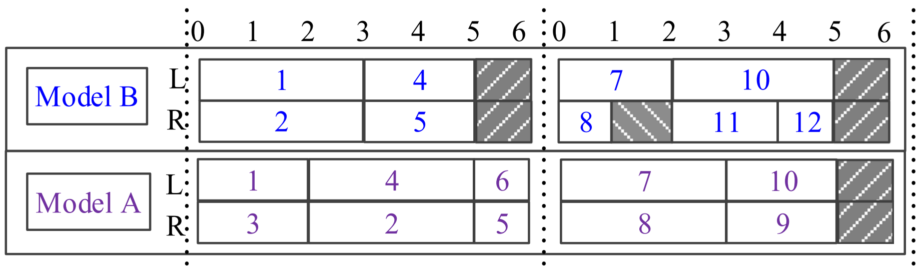


Fig. 1 Task allocation in a mixed-model two-sided assembly line

$C(i)$	Set of tasks whose operation directions are opposite to that of task i; $C(i) = \begin{cases} AL & \text{if } i \in AR \\ AR & \text{if } i \in AL \\ \phi & \text{if } i \in AE \end{cases}$
t_{im}	Operation time of task i of model m
ψ	A very large positive number
CT	Cycle time
w_1, w_2	The weights of an opened mated-station and an opened workstation
Decision variables	
x_{ijk}	1, if task i is assigned to side k of mated-station j; 0, otherwise
tf_{im}	Completion time of task i of model m
G_j	1, if one side of mated-station j is utilized; 0, otherwise
F_j	1, if both sides of mated-station j are utilized; 0, otherwise
v_{jk}	1, if workstation (j, k) is utilized; 0, otherwise
Indicator variables	
z_{ip}	1, if task i is assigned earlier than task p at the same workstation; 0, if task p is assigned earlier than task i at the same workstation

On the basis of Özcan and Toklu [18], the mathematical model is formulated utilizing expressions (1–10). The objective in expression (1) minimizes the number of mated-stations and the number of workstations. Constraint (2) is the occurrence constraint, indicating that each task must be allocated to exactly one side of a mated-station. Constraint (3) and (4) deal with the cycle time constraint: constraint (3) indicates that the completion time of each task for each model must be less than or equal to the cycle time; constraint (4) indicates that the completion time of each task for each model must be larger than or equal to the corresponding operation time. Constraint (5) and (6) deal with the precedence constraint: constraint (5) indicates that the predecessors of one task must be allocated to the former or the same mated-station; constraint (6) denotes that the predecessor of one task must be completed before operating this task when the predecessor and this task are allocated to the same mated-station. Constraint (7) and (8) regard the situation when two tasks have no precedence relationship and they are allocated to the same workstation: constraint (7) is reduced to $tf_{pm} - tf_{im} \geq t_{pm}$ when task i is operated before task p; constraint (8) is reduced to $tf_{im} - tf_{pm} \geq t_{im}$ when task p is operated before task i. Constraint (9) indicates that v_{jk} is equal to 1 when there is at least one task in the combined precedence relation allocated to workstation (j, k). Lastly, constraint (10) determines the values of G_j and F_j.

$$\text{Minimize} \quad w_1 \cdot \sum_{j \in J} \left(F_j + G_j\right) + w_2 \cdot \sum_{j \in J} \sum_{k=1,2} v_{jk} \tag{1}$$

$$\sum_{j \in J} \sum_{k \in K(i)} x_{ijk} = 1 \forall i \in I \tag{2}$$

$$tf_{im} \leq CT \quad \forall i \in I, m \in M \tag{3}$$

$$tf_{im} \geq t_{im} \forall i \in I, m \in M \tag{4}$$

$$\sum_{g \in J} \sum_{k \in K(h)} g \cdot x_{hgk} \leq \sum_{j \in J} \sum_{k \in K(i)} j \cdot x_{ijk} \quad \forall i \in I - P_0, h \in P(i) \tag{5}$$

$$tf_{im} - tf_{hm} + \psi\left(1 - \sum_{k \in K(h)} x_{hjk}\right) + \psi\left(1 - \sum_{k \in K(i)} x_{ijk}\right) \geq t_{hm}, \\ \forall i \in I - P_0, h \in P(i), j \in J, m \in M \tag{6}$$

$$tf_{pm} - tf_{im} + \psi\left(1 - x_{ijk}\right) + \psi\left(1 - x_{pjk}\right) + \psi\left(1 - z_{ip}\right) \geq t_{pm} \quad \forall i \in I, m \in M, \\ p \in \left\{r | r \in I - \left(P_a(i) \cup S_a(i) \cup C(i)\right) \text{ and } i < r\right\}, j \in J, k \in K(i) \cap K(p) \tag{7}$$

$$tf_{im} - tf_{pm} + \psi\left(1 - x_{ijk}\right) + \psi\left(1 - x_{pjk}\right) \\ + \psi \cdot z_{ip} \geq t_{im} \quad \forall i \in I, m \in M, \\ p \in \left\{r | r \in I - \left(P_a(i) \cup S_a(i) \cup C(i)\right) \text{ and } i < r\right\}, \\ j \in J, k \in K(i) \cap K(p) \tag{8}$$

$$\sum_{i \in I} x_{ijk} - \psi \cdot v_{jk} \leq 0 \quad \forall j \in J, k \in \{1, 2\} \tag{9}$$

$$\sum_{k=1,2} v_{jk} - 2 \cdot F_j - G_j = 0 \quad \forall j \in J \tag{10}$$

4 Encoding and decoding

This section introduces the encoding schemes in Sect. 4.1 and the decoding schemes in Sect. 4.2, and thereafter compares the tested decoding schemes in Sect. 4.3.

4.1 Encoding schemes

The encoding schemes for the MTALBP-I are similar to that for the TALBP-I, and this study mainly tests two encoding schemes: (1) permutation-oriented encoding where only task permutation is employed and (2) permutation-oriented and side-oriented encoding where task permutation and side vector are employed. In the first method, directions of E-type tasks are determined in the decoding process utilizing the heuristic method. The second method utilizes the task side vector to determine the directions before the decoding. For the first permutation-oriented encoding, a task permutation is utilized to encode all the tasks. The tasks in the former positions of the task permutation are allocated first. A possible task permutation, for the example in Fig. 1, is {1, 3, 2, 4, 5, 6, 8, 7, 9, 10, 11, 12}. For instance, task 1 is in the first position of the task permutation and has the highest priority, and it should thus be allocated first when all the constraints are satisfied. The task permutation does not provide detailed information about the allocated sides of E-type tasks, and the directions of E-type tasks are determined in the decoding process. For the second method, a side vector is applied, and the example corresponding to Fig. 1 is {L, R, R, L, R, L, L, R, R, L, R, R}. The first position is L or left, denoting that task 1 is allocated to the left side.

4.2 Decoding schemes

The decoding procedures for the MTALBP-I are quite different from those utilized in solving the TALBP-I, as all the tasks for the models in the MTALBP-I must be completed in the given cycle time. The general decoding procedure for the MTALBP-I is presented in Algorithm 1. It is to be noted that the detailed information for dealing with the MTALBP-I is omitted due to space constraints, however, in the MTALBP-I the models need to be considered separately. For instance, a task is assignable in Step 2 when the completion time of this task for each model is less than or equal to the cycle time. In Step 5, the tasks of the last mated-station can be removed to one workstation only when the total operation time for each model on the last mated-station is not larger than the cycle time.

All the reported research on this topic have the same Step 1, Step 2, and Step 4, though the methods for selecting a task and a side in Step 3 are different. All the reported methods for the selection of tasks and sides, along with the proposed new method, are summarized here. Among them, the first eight methods utilize only task permutation for decoding whereas the ninth method utilizes both task permutation and task side vector.

(1) *Task-to-workstation decoding 1 or TSD1* An assignable task in the former position is first selected. If this task can be allocated to only one side, the corresponding side is selected. Namely, the L-type/R-type tasks are allocated to the left/right side respectively. If the selected task can be allocated to either side, a side is randomly selected as the current workstation.
(2) *Task-to-workstation decoding 2 or TSD2* An assignable task in the former position is first selected. If this task can be allocated to only one side, the corresponding side is selected. If the selected task can be allocated to either side, the left side is selected by default as the current workstation.
(3) *Task-to-workstation decoding 3 or TSD3* An assignable task in the former position is first selected. If this task can be allocated to only one side, the corresponding side is selected. If the selected task can be allocated to either side, the side with a larger capacity is selected

```
Algorithm 1. Decoding procedure for the MTALBP-I
Step 1: If some tasks are still unallocated, open a new mated-station. Otherwise, go to Step 5.
Step 2: Check whether assignable tasks exist. If no tasks are assignable, go to Step 1.
% A task is assignable when it has not been allocated and it satisfies the precedence constraint, cycle time constraint, and direction constraint.
Step 3: Select an assignable task and a side of the current mated-station and allocate this task to the selected side.
Step 4: Update the remaining capacities for the two sides of the current mated-station and go to Step 1.
Step 5: Remove the tasks of the two sides of the last mated-station to one side when satisfying the following conditions: 1) the two sides are both utilized; 2) the total operation time is smaller than or equal to the cycle time; 3) the tasks' directions are not conflicted.
```

as the current workstation or a random side is selected when the capacities for both sides are equal.

(4) *Task-to-workstation decoding 4 or TSD4* An assignable task in the former position is first selected. If this task can be allocated to only one side, the corresponding side is selected. If the selected task can be allocated to either side, the side with a larger capacity is selected as the current workstation or the left side is selected when the capacities for both sides are equal.

(5) *Workstation-to-task decoding 1 or STD1* Both sides of the current mated-station are checked for whether it is possible to allocate the tasks to them. If only one side is available for allocated tasks, this side is selected. If both sides are available, the side with a larger capacity is selected or a random side is selected when both sides have the same capacities. Subsequently, an assignable task in the former position of the task permutation is selected to be allocated to this selected side.

(6) *Workstation-to-task decoding 2 or STD2* Both sides of the current mated-station are checked for whether it is possible to allocate the tasks to them. If only one side is available for allocated tasks, this side is selected. If both sides are available, the side with a larger capacity is selected or the left side is selected by default when both sides have the same capacities. Subsequently, an assignable task in the former position of the task permutation is selected to be allocated to this selected side.

(7) *Workstation-to-task decoding 3 or STD3* Both sides of the current mated-station are checked for whether it is possible to allocate the tasks to them. If only one side is available for allocated tasks, this side is selected. If both sides are available, the side with a larger capacity is selected or a random side is selected when both sides have the same capacities. During the task selection process, we first obtain all the assignable tasks and check whether the tasks which can be operated at the earliest possible time of the selected workstation exist. If so, the tasks which cannot be operated at the earliest possible time are deleted from the assignable task set. Lastly, a task in the former position of the task permutation is selected.

(8) *Workstation-to-task decoding 4 or STD4* Both sides of the current mated-station are checked for whether it is possible to allocate tasks to them. If only one side is available for allocated tasks, this side is selected. If both sides are available, the side with a larger capacity is selected or the left side is selected by default when both sides have the same capacities. During the task selection process, we first obtain all the assignable tasks and check whether the tasks which can be operated at the earliest possible time of the selected workstation exist. If so, the tasks which cannot be operated at the earliest possible time are deleted from the assignable task set. Lastly, a task in the former position of the task permutation is selected.

(9) *Two vector decoding or TVD* The assignable sides for all E-type tasks are obtained on the basis of the side vector. Then, an assignable task in the former position is first selected and allocated to the corresponding side. It is to be noted that in the former eight decoding procedures, E-type tasks can be allocated to either side, but in this decoding scheme the side for a task is determined in advance.

Note that the detailed information for dealing with the MTALBP-I is not exhibited here due to space constraints, though the detailed decoding procedures of *TSD4* is presented in "Appendix A" as an example. However, all the detailed decoding procedures and corresponding information are available upon request, where the models must be considered separately. Among these decoding methods, only three methods have been utilized in the literature (TSD1, TSD3, and TSD4), and the remaining six decoding schemes are developed to solve the MTALBP for the first time. Notice that, as can be seen in Sect. 6.2, the different decoding schemes have a great impact on the final performance of the algorithms. However, the above finding does not receive enough attention and a comparative study to evaluate these decoding schemes has not yet been undertaken. Hence, this

Table 2 The complexity comparison among decoding schemes

Decoding scheme	Number of possible solutions	Source
TSD1	$[Nt!, 2^{Ne} \cdot Nt!]$	Özcan and Toklu [18]
TSD2	$Nt!$	Developed for the first time in this paper
TSD3	$[Nt!, 2^{Ne} \cdot Nt!]$	Delice et al. [5]
TSD4	$Nt!$	Yuan et al. [27]
STD1	$[Nt!, 2^{Ne} \cdot Nt!]$	Developed for the first time in this paper
STD2	$Nt!$	Developed for the first time in this paper
STD3	$[Nt!, 2^{Ne} \cdot Nt!]$	Modified from Khorasanian et al. [7]
STD4	$Nt!$	Modified from Li et al. [16]
TVD	$2^{Ne} \cdot Nt!$	Developed for the first time in this paper

study presents all the decoding schemes here and conducts the comparative study in Sect. 6.2 in order to evaluate these decoding schemes for the first time, with the purpose of providing useful information and guidance for future research.

4.3 Decoding scheme comparisons

Based on the literature, it can be seen that there are $Nt!$ possible task permutations for a type I one-sided assembly line if the task permutation-oriented encoding is utilized, where Nt is the number of tasks and each task permutation corresponds to a solution and there are $Nt!$ possible solutions. However, in the case of the MTALBP-I, there will be $Nt!$ possible task permutations, where each task permutation might correspond to many solutions due to the existence of E-type tasks. This situation results in $2^{Ne} \times NT!$ possible solutions, where N_e is the number of E-type tasks.

The nine decoding procedures have different search spaces due to different approaches to handling E-type tasks, and are summarized in Table 2. For TVD, the number of all possible task permutations is $Nt!$, though the E-type tasks need to be tested on both the sides, which results in $2^{Ne} \cdot Nt!$ possible solutions. For the remaining methods, TSD2 allocates the E-type task to the left side by default. In this case, E-type tasks are not required to be tested on both sides, which reduces the number of solutions of $Nt!$. This situation also suits TSD4, STD2, and STD4, which utilize heuristic methods to determine the selected side for E-type tasks. TD1 selects a side randomly for E-type tasks, and each task permutation might correspond to many solutions. However, in some instances E-type tasks can be allocated to only one side and thus the number of possible solutions is less than or equal to $2^{Ne} \cdot Nt!$. This situation also suits TSD3, STD1, and STD3, where a random side is selected in some situations. In addition, this paper provides only the ranges for TSD1, TSD3, STD1, and STD3 since, to the authors' best knowledge, the real number of possible solutions is still unknown. It is worth noting that the search space of TSD3 is much smaller than that of TSD1, since TSD3 allocates the E-type tasks to the side with a larger capacity when both sides have different capacities.

Table 2 presents the sources of these decoding schemes, and these decoding schemes cover all the reported instances for the MTALBP-I. TSD1 is the first decoding reported, and it ignores the balance of workloads and reduction of sequence-dependent idle time. TSD3 and TSD4, on the contrary, allocate the tasks to the side with a larger capacity so as to obtain a well-balanced solution. STD1 and STD2 also improve the balance of the obtained solutions by allocating tasks to the side with a larger remaining capacity. STD3 and STD4 take into account the sequence-dependent idle time during the decoding process. These methods prioritize the tasks which can be operated at the earliest possible time of the selected workstation and aim at reducing sequence-dependent idle time. TVD determines a solution on the basis of task permutation and task side vector, and the final solution depends only on the update of these two vectors. It is worth pointing out that the optimality may not be achievable

Fig. 2 Pseudo-code of the presented IG

```
Procedure of IG ( $d, T$ )
  $\pi$ := Modified NEH-based initialization;                    %Initialization
  $\pi$:=Local search($\pi$ )                                     %Local search
  $\pi_b := \pi$ ;                                                 %Update best solution
  While (termination criterion is not satisfied) do
    $\pi' := \pi$;                                                %Destruction phase
    For $i$:=1 to $d$ do
        $\pi'$ := remove one task of $\pi'$randomly to position $Nt$-$d$+$i$ ;
    Endfor
    For $i$:=1 to $d$ do                                         %Construction phase
        $\pi'$ :=best permutation by inserting task $\pi_{Nt-d+i}'$ in all possible position;
    Endfor
    $\pi''$:=Local search($\pi'$)                                 %Local search
    If $Fit(\pi'') < Fit(\pi)$then                                % Acceptance criterion
        $\pi := \pi''$ ;
        If $Fit(\pi'') < Fit(\pi_b)$then                          %Update best solution
            $\pi_b := \pi''$ ;
        Endif
    Elseif $rand(0,1) \le \exp\{-(Fit(\pi'') - Fit(\pi) / Temperature\}$ then
        $\pi := \pi_b$ ;
    Endif
  Endwhile
```

if one uses the first eight decoding schemes in some cases, since they all utilize heuristic methods to determine the sides of E-type tasks. Notice that this paper considers allocating the E-type tasks to the left side by default for TSD2, TSD4, STD2, and STD4, though allocating the E-type tasks to the right side is also practical. Actually, the left and the right side are both tested in the preliminary experiments, and they show similar performance. A detailed comparison campaign of the nine decoding schemes is presented in Sect. 6.2.

5 Local search methods

Different optimization algorithms have been applied to solve the TALBP, but some are too complex or difficult to extend to solve other variants of the TALBP. To overcome this concern, the main focus of this paper is to develop simple methods with high performance. The simple and effective iterated greedy (IG) algorithm [23] and iterated local search (ILS) algorithm [21] are employed and modified to solve the proposed MTALBP-I. Both IG and ILS are simple stochastic methods that have demonstrated good performance in optimization problems despite their simplicity of implementation.

5.1 Iterated greedy algorithm

The proposed IG starts with constructing a high-quality initial solution and improving this initial solution using a local search. Then, the following four steps repeat interactively: the destruction, construction, local search, and acceptance. The procedure of the implemented IG algorithm is depicted in Fig. 2. This paper utilizes a modified NEH-based initialization for the initialization process presented in Li et al. [17], which can obtain a high-quality solution. The local search is employed to emphasize intensification, which is presented in detail in Sect. 5.3. Within the iteration, the destruction phase destructs the current individual by moving d randomly selected tasks to the ending positions of the current permutation π. The construction phase is applied to improve the new permutation $\pi\prime$ by inserting these d tasks into all the possible positions. It is to be noted that the proposed destruction phase and construction phase are different from those used in Ruiz and Stützle [23]. d random tasks are not removed from permutation $\pi\prime$ but are inserted into the backward positions, guaranteeing that each task permutation is able to acquire a feasible solution. This modification is carried out due to the difficulty in evaluating part of the task permutation. Subsequently, a local search is also applied to enhance this new task permutation $\pi\prime$. Finally, the acceptance criterion is utilized to determine whether this new permutation $\pi\prime\prime$ can replace the incumbent permutation π. If the new permutation outperforms the incumbent one, it replaces the incumbent one. Otherwise, it replaces the incumbent one with a possibility of $\exp\{-(Fit(\pi\prime\prime)) - Fit(\pi)/Temperature\}$. This acceptance criterion allows the acceptance of a worse solution. Both d and $Temperature$ are important parameters to select, and they need to be carefully calibrated to achieve good results.

5.2 Iterated local search algorithm

ILS is also a simple local search algorithm proposed by Pan and Ruiz [21] which shows promising results for different types of optimization problems. It is adopted to solve this problem since it is easy to implement and quite effective and

Fig. 3 Pseudo-code of the presented ILS

```
Procedure of ILS ( $d$, $nm\_move$ )
  $\pi$ := Modified NEH-based initialization;      %Initialization
  $\pi$ :=Local search($\pi$)                        %Local search
  Repeat
    $\pi' := \pi$;
    For $i$=1 to $nm\_move$ do                        %Perturbation
      For $j$=1 to $d$ do
        Select a random task from $\pi'$ and insert it into a random position;
      Endfor
    End for
    $\pi''$:= Best individual among $nm\_move$ perturbed solutions $\pi'$;
    $\pi''$:=Local search($\pi''$)                    %Local search
    If $Fit(\pi'') \le Fit(\pi)$ then                 % Acceptance criterion
      $\pi := \pi''$;
    Endif
  Until termination criterion is met
```

Fig. 4 The procedure of new precedence-based local search

```
Procedure Local search ($\pi, \pi^{rp}, a$)
 counter:=0;i:=1;
While (counter<$Nt \times Nt/a$ )
    for j:=1 to b do
      If (Rand()%100<50)                                   % Insert operation
         Remove task$\pi_i^{rp}$ from $\pi$ ;
         $\pi$':= Permutation by inserting task$\pi_i^{rp}$ into a random position (without repetition);
      Else                                                 %Swap operation
         Select a random task and exchanges the positions of this task and $\pi_i^{rp}$;
      Endif
      If (New task permutation is violated to precedence constraints) %Precedence check
         Continue;
      Else
         Obtain a feasible solution based on the new task permutation;
      Endif
      If Fit($\pi$')<Fit($\pi$);
         $\pi$ :=$\pi$'; counter:=0;
      Elseif Fit($\pi$ )=Fit($\pi$');
            $\pi$ :=$\pi$';
      Endif
    Endfor
    counter:=counter+1;
    i:=mod(i+1, nt)
 Endwhile
```

efficient. The procedure of the implemented ILS is presented in Fig. 3. Like IG, ILS starts by generating a high-quality initial solution and improving the initial solution using a local search. Then, perturbation, local search, and acceptance criterion application are performed in a loop which is executed until a termination criterion is satisfied. In the perturbation phase, *nm_move,* new permutation $\pi\prime$ is generated by implementing a randomly inserted operator d times on the current permutation π, and then the best individual is selected. This perturbation procedure is easier and simpler than the destruction phase and construction phase in the IG algorithm. Subsequently, a local search is applied to enhance the new permutation $\pi\prime\prime$, and acceptance criterion is applied to decide whether this new solution can be accepted. As for the acceptance criterion, the ILS adopts a simpler method by only accepting a solution with an equal or better objective function value. The main idea behind this modification is to develop an easier method. By doing so, this modification does not cause a large difference in the algorithm performance with preliminary experiments. *nm_move* and d are two important parameters that need to be calibrated. The proposed ILS algorithm can be considered an easier method than the IG algorithm.

5.3 Improved local search procedure

The local search procedure plays an important role in the performance of IG and ILS methods. This paper proposes an improved precedence-based local search with two neighborhood structures and referenced permutation. The proposed local search procedure is depicted in Fig. 4. The local search procedure aims to optimize the current task permutation π, and this local search terminates when an improved permutation has not been achieved for a number of consecutive $Nt \times Nt/a$ iterations, where a is a parameter that must be calibrated. Within a cycle, the task π_i^{rp} is selected from a referenced permutation π^{rp}, and this task is either inserted into another position or exchanged with another random task. After executing the insert operator or swap operator, the new task permutation π' is checked to ensure that the precedence constraint is satisfied. If the precedence constraint is not violated, a new solution is obtained using this new task permutation. If the new fitness value is better than the current one, then the incumbent task permutation is replaced with the new one and the value of the *counter* is set to zero. If the new fitness is equal to the current one, the incumbent permutation is also updated. In this paper, the referenced permutation is set to be the same as the current best permutation.

This local search has several features leading to faster computation and an effective search capacity. Specifically

(1) this referenced permutation ensures that each task undergoes a local search and reduces the possibility of updating the positions of a same task again and again. (2) Both the insert operator and swap operator are combined with the aim of increasing the search space. (3) Another important feature in the improved local search procedure is that the position of the selected task is modified *a* times, rather than testing the selected task on all the possible positions, as is done in Ruiz and Stützle [23]. This also assists in increasing the search speed. Moreover, the search speed is further accelerated by testing only task permutations satisfying the precedence constraint. From the preliminary results, this new local search could demonstrate better performance than that proposed by Li et al. [16].

5.4 Two-stage evaluation procedure

Based on Li et al. [15], this study develops a two-stage evaluation procedure with two objectives in expressions (11–13). Here, Nm and Ns are the number of mated-stations and the number of workstations, and w_1 and w_2 are the corresponding weights. Indexes j, k, and m denote a task, a side of a mated-station, and a model, respectively. ST_{jkm} is the workload (the total operation time of tasks on workstation (j,k)) allocated to workstation (j,k) for model m, and CT is the cycle time. WSI is a weighted smoothness index calculated using expression (13) and WSI_0 is the weighted smoothness index obtained from the initial solution. WL_{jkm} is the finishing time of workstation (j,k) for model m and WL_{max} is the maximum value of WL_{jkm}.

$$\text{Min Fitness1} = w_1 \times Nm + w_2 \times Ns + \frac{\sum_{m\in M}\sum_{j\in J}\sum_{k=1,2} q_m \times (Nm + 1 - j) \times \left(CT - ST_{jkm}\right)}{2 \times Nm \times CT} \tag{11}$$

$$\text{Min Fitness2} = w_1 \times Nm + w_2 \times Ns + WSI/WSI_0 \tag{12}$$

$$WSI = \sqrt{\frac{\sum_{m\in M} q_m \times \left(\sum_{j\in J}\sum_{k=1,2}\left(WL_{jkm} - WL_{\max}\right)^2\right)}{Ns}} \tag{13}$$

In this study, the two-stage evaluation procedure first utilizes the first evaluation objective, referred to as station-oriented evaluation, in expression (11), and later utilizes the second evaluation objective in expression (12) when the optimal solution in terms of the number of mated-stations and the number of workstations is achieved. The first evaluation objective in expression (11) aims at optimizing the mated-station number and workstation number by selecting the solution with more allocated workloads to the former mated-stations. In expression (11), the idle time of the former mated-station is provided with a larger weight, and thus the solution with more workload on former workstations is preserved. Nevertheless, the first evaluation objective might result in unbalanced workloads on workstations, where the former mated-stations have much larger workloads. Hence, the second evaluation objective in expression (12) is utilized here only when the first evaluation objective cannot further reduce the workstation number or a termination criterion for the first stage is satisfied. Expression (12) is used to minimize the weighted smoothness index to preserve the solutions with balanced workloads, aiming at optimizing the balance of workloads. Since a mated-station comprises two workstations, the w_1 and w_2 are set to 2 and 1 respectively. As the latter part of expression (11) or expression (12) is usually smaller than 1.0, this part takes effect only when solutions have the same mated-stations and workstations.

It might be augured that utilizing only expression (12) is sufficient, though only utilizing expression (12) might obtain poor performance; the reasons for utilizing the two-stage evaluation procedure are analyzed as follows. Among the reported papers, there are two types of objectives solved in the MTALBP-I: minimizing the $w_1 \times Nm + w_2 \times Ns$ in Delice et al. [5] and maximizing the line efficiency and minimizing the smoothness proposed by Özcan and Toklu [18]. Since there are many solutions having the same number of mated-station and workstations, the $w_1 \times Nm + w_2 \times Ns$ cannot determine the better one and thus cannot determine the proper evolutionary direction. Özcan and Toklu [18] objective is also not effective since it might lose the ability to find the optimal solution. If the workloads on workstations are balanced, it is difficult or even impossible to reduce the number of workstations with a small adjustment. Nevertheless, if there are fewer workloads on the latter workstations, there is a high probability of reducing the number of workstations with a small adjustment. In short, the two published objectives might not be capable of reducing the number of mated-stations and workstations effectively. As will be seen in Sect. 6.2, this station-oriented evaluation in expression (11) outperforms the other two published objectives by a significant margin in terms of reducing the number of workstations.

6 Computational results

This section presents the experimental design and computational results. Tested benchmark datasets and the adaptions of eleven other metaheuristic algorithms are explained in Sect. 6.1. The performances of the nine decoding schemes

Table 3 Description of tested benchmark problems

Problem	Number of tasks	Number of models	Number of tested cases	Tested cycle times
P9	9	2	3	4, 5, 6
P12	12	2	4	5, 6, 7, 8
P16	16	2	6	15, 16, 18, 19, 21, 22
P24	24	2	6	20, 24, 25, 30, 35, 40
P65	65	3	5	326, 381, 435, 490, 544
P148	148	4	7	204, 255, 306, 357, 408, 459, 510
P205-Yuan	205	5	10	1133, 1322, 1510, 1699, 1888, 2077, 2266, 2454, 2643, 2832
P205-Delice	205	5	10	1133, 1322, 1510, 1699, 1888, 2077, 2266, 2454, 2643, 2832

and several objectives are compared in Sect. 6.2. The computational and statistical results are presented in Sect. 6.3.

6.1 Experimental design

To evaluate the proposed algorithms, most of the available benchmark problems, to the authors' best knowledge, are solved: four small-sized problems (P9, P12, P16, and P24) and four large-sized problems (P65, P148, P205-Yuan, and P205-Delice). The precedence relations and operation directions of P9, P12, and P24 are taken from Kim et al. [8], and those of P16, P65, P205-Yuan, and P205-Delice are taken from Lee et al. [13]. The precedence relations and operation directions of P148 are taken from Bartholdi [2]. The task times of the P9, P12, P16, P24, P65, and P148 are taken from Özcan and Toklu [18], the task times of P205-Yuan are taken from Yuan et al. [27], and the task times of P205-Delice are taken from Delice et al. [5]. The benchmark problems are summarized in Table 3. The overall proportions of all models are assumed to be the same, namely $q_A = q_B = \ldots = q_m$ [18].

To test the performance of the proposed algorithms, this research presents several adaptions of other well-known and recent metaheuristic algorithms. Eleven algorithms are re-implemented, among which nine algorithms are developed for the first time to solve the MTALBP-I. The compared methods include a genetic algorithm (GA), ant colony optimization algorithm (ACO), simulated annealing algorithm (SA), tabu search algorithm (TS), two-ant colony optimization algorithm (2ACO), bee optimization algorithm (BA), particle swarm optimization with negative knowledge (PSONG), particle swarm optimization algorithm (PSO), teaching–learning-based optimization algorithm (TLBO), late acceptance hill-climbing algorithm (LAHC), and discrete artificial bee colony algorithm (DABC) [15]. During the re-implementing process, some adaptions are necessary, including adopting the provided new objective and effective decoding scheme. Due to space constraints, the pseudo-algorithms of these methods are not presented, but the basic information of these algorithms is available upon request. All the algorithms are re-programmed utilizing the C++ language in Microsoft Visual Studio 2012, and they utilize the same termination criteria of an elapsed CPU time of $Nt \times Nt \times \rho$ milliseconds. To avoid prejudiced comparison, ρ is set to 5, 10, 15, and 20, respectively, to analyze the performance of the tested methods for short to very large computational times. In the termination criterion, the large-sized problems consume more computational time during solution search. All the experiments are tested on a set of personal computers with the same setting, namely equipped with an Intel Core2 2.33GHZ CPU and 3.036 GB memory.

To highlight the effectiveness of the proposed station-oriented evaluation in expression (11) as well as the rationality of utilizing the two-stage evaluation procedure, the following comparison in Sects. 6.2 and 6.3 focuses mainly on the results in terms of the number of mated-stations and the number of workstations. After conducting the experiments, it is observed that $Nm = Ns/2 + Ns\%2$ for all the tested cases. In other words, the value of Nm can be achieved

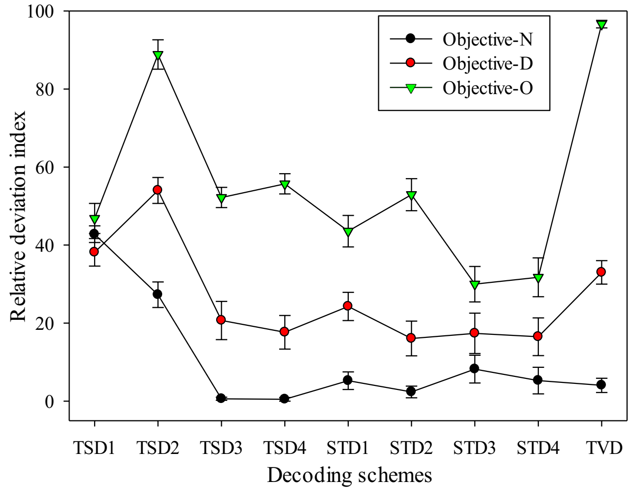


Fig. 5 Means plot of decoding schemes with three objectives

directly once the value of the *Ns* is given. For simplicity, this section mainly presents the results in terms of the number of workstations. Notice that the results in terms of the number of mated-stations can be achieved utilizing the above expression, and are also available upon request.

6.2 Computational analysis of decoding schemes and objectives

This section presents the results in terms of the number of workstations utilizing nine decoding schemes and three objectives to show their performance regarding optimizing the number of the workstations. The three objectives comprise the station-oriented evaluation in expression (11) (referred to as *Objective-N*), the objective taken from Özcan and Toklu [18] (referred to as *Objective-O*), and the objective from Delice et al. [5] (referred to as *Objective-D*). *Objective-O* optimizes the maximization of the line efficiency and minimization of the workload smoothness at the same time, while *Objective-D* minimizes the number of mated-stations and the number of workstations. Note that *Objective-O* is tested to verify the rationality of utilizing the two-stage evaluation procedure in sequence, and *Objective-D* is tested to prove the superiority of the station-oriented evaluation as the first evaluation objective in terms of optimizing the mated-station and workstation numbers.

This study utilizes the SA algorithm to test the decoding schemes and objectives. Ten cases of P205-Yuan are solved, and the average results of 10 repetitions are recorded for the four termination criteria. Since different cases are solved, the relative deviation index (RDI) is selected to transfer the achieved workstation numbers using expression (14), where Fit_{some} is the workstation number of one case through a combination of decoding scheme and objective and Fit_{worst} and Fit_{best} are the largest and smallest workstation number of the same case among all the combinations.

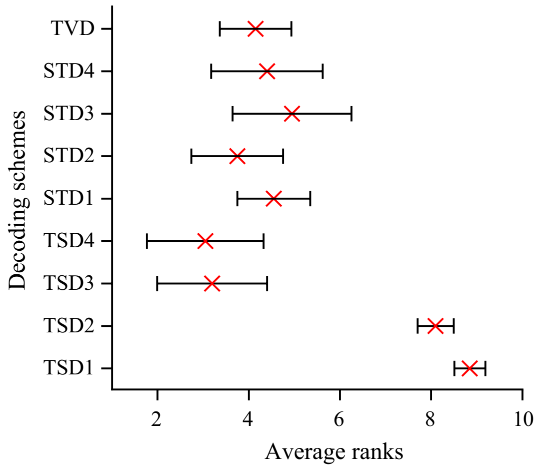


Fig. 6 Means plot of the average ranks and 95% confidence intervals for nine decoding schemes

solution with fewer workstations with a small adjustment. On the contrary, there are much larger possibilities to reduce the workstation number of a solution with little workload on the last workstation.

In summary, this comparative study proves the following. (1) The line efficiency and smoothness should be optimized in sequence as the simultaneous optimizing of the line efficiency and smoothness leads to poor results regarding the mated-station number and the workstation number. (2) The proposed station-oriented evaluation, as the first evaluation objective, outperforms other objectives in terms of optimizing the mated-station number and workstation number, and is a good option for utilizing the two-stage evaluation procedure to optimize the line efficiency and line smoothness in a sequence.

Due to the clear superiority of the proposed *Objective-N* in terms of the number of workstations as presented in

$$\text{Relative Deviation Index(RDI)} = 100 \times \left(Fit_{some} - Fit_{best}\right) / \left(Fit_{worst} - Fit_{best}\right) \tag{14}$$

The mean RDI values of different decoding schemes and objectives are depicted in Fig. 5 under $\rho = 20$. It can be noted that the proposed *Objective-N* shows a clear advantage over the two compared schemes, whereas the performances of decoding schemes are different for the three objectives. The best results are obtained by *Objective-N* and followed by *Objective-D* and *Objective-O*. It is clear that *Objective-N* performs best. However, it is interesting to note that *Objective-D* performs better than *Objective-O*. This could be due to the fact that *Objective-O* optimizes the line efficiency and workload balance together, leading to a possibility of losing optimal solutions. In general, it is very difficult or even impossible to transfer a well-balanced solution into a new

Fig. 5, the main focus is on the decoding scheme under the condition of *Objective-N*. The non-parametric Friedman rank-based analysis is utilized to analyze the results obtained by the different decoding schemes due to the deviation from normality. Since there are nine decoding schemes, the results of the algorithms for each case are transferred so that the smallest value is ranked 1 and the largest value is ranked 9. Analysis results indicate that there is a statistical difference in the average ranks of the decoding schemes with a *P* value lower than 0.01. The mean plot of the ranks for the decoding schemes under $\rho = 20$ is depicted in Fig. 6. In addition, it is observed that TSD4 ranks first, TSD3 ranks second, and finally TSD1 ranks ninth.

Table 4 Comparison of algorithms with four termination criteria

Problem	GA	DABC	BA	TLBO	PSO	LAHC	SA	TS	PSONG	ACO	2ACO	IG	ILS
$\rho = 5$													
P9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P24	38.6	39.3	45.0	19.1	65.1	20.2	28.6	33.2	19.8	100.0	63.9	3.9	25.9
P65	57.2	56.5	56.6	58.4	66.1	49.9	47.5	28.5	57.0	100.0	68.4	41.0	38.4
P148	25.4	27.3	23.8	51.5	55.8	10.5	21.3	14.6	57.6	100.0	29.8	12.8	15.5
P205-Yuan	4.6	5.3	6.0	11.4	13.7	2.7	2.3	1.8	10.0	100.0	43.6	0.5	0.9
P205-Delice	20.9	20.0	18.5	28.6	30.6	11.7	8.0	10.5	30.2	100.0	46.8	4.9	4.0
Avg	18.6	18.9	18.9	22.9	30.5	11.5	13.0	11.1	23.7	74.5	36.0	7.3	9.9
$\rho = 10$													
P9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P24	35.1	37.3	28.1	15.3	65.1	19.1	26.5	30.1	19.8	94.6	63.9	1.7	24.6
P65	54.2	49.7	52.9	56.0	66.1	48.5	36.2	25.3	54.0	98.1	61.7	23.8	37.3
P148	20.5	26.1	17.2	47.1	55.7	6.5	16.8	9.5	30.6	97.1	29.2	7.8	10.3
P205-Yuan	3.2	3.1	3.8	9.4	13.7	1.8	0.9	0.7	8.4	100.0	43.6	0.2	0.3
P205-Delice	14.2	8.8	12.3	28.0	30.5	10.6	7.7	9.4	27.7	100.0	46.8	2.1	2.7
Avg	15.7	15.2	14.0	21.1	30.5	10.3	10.7	9.3	18.9	73.3	35.3	4.1	8.5
$\rho = 15$													
P9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P24	30.4	36.3	25.9	14.7	65.1	18.1	26.5	30.1	19.8	91.9	63.6	0.3	21.8
P65	50.4	47.4	49.7	55.2	66.1	46.1	35.5	22.8	53.8	98.1	59.1	17.7	36.5
P148	19.3	25.3	13.2	46.9	55.7	4.7	13.0	6.9	27.6	92.1	28.9	2.6	3.5
P205-Yuan	2.7	2.0	3.1	9.1	13.7	1.3	0.6	0.6	7.3	100.0	43.1	0.1	0.1
P205-Delice	12.7	6.5	10.7	27.8	30.5	9.7	6.8	8.5	26.0	100.0	46.8	1.7	1.1
Avg	14.2	14.1	12.4	20.8	30.5	9.5	9.8	8.5	17.9	72.3	34.9	2.5	6.9
$\rho = 20$													
P9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P24	28.3	35.0	22.7	14.5	65.1	18.1	25.5	30.1	19.8	89.2	63.5	0.0	21.2
P65	49.7	45.7	46.3	54.2	66.1	37.5	34.6	22.7	53.7	98.1	59.0	0.0	16.4
P148	17.9	23.7	11.7	45.4	54.8	4.6	9.4	6.7	27.0	90.7	28.6	1.1	1.1
P205-Yuan	2.3	1.4	2.7	8.8	13.7	1.2	0.5	0.5	6.2	100.0	43.1	0.1	0.0
P205-Delice	11.1	6.4	9.9	27.5	30.5	9.6	6.2	8.3	24.0	100.0	46.7	1.5	0.9
Avg	13.3	13.4	11.3	20.4	30.3	8.6	9.0	8.4	17.2	71.8	34.8	0.5	4.4

6.3 Comprehensive comparison of metaheuristics

This section presents the experimental results obtained from all the tested methods. It is to be noted that there are nine selected decoding schemes and three selected objectives, leading to 27 configurations for each algorithm. Due to space constraints, this section presents only the computational results obtained using decoding scheme TS4 and *Objective-N*, which is proved to be the most effective in Sect. 6.2.

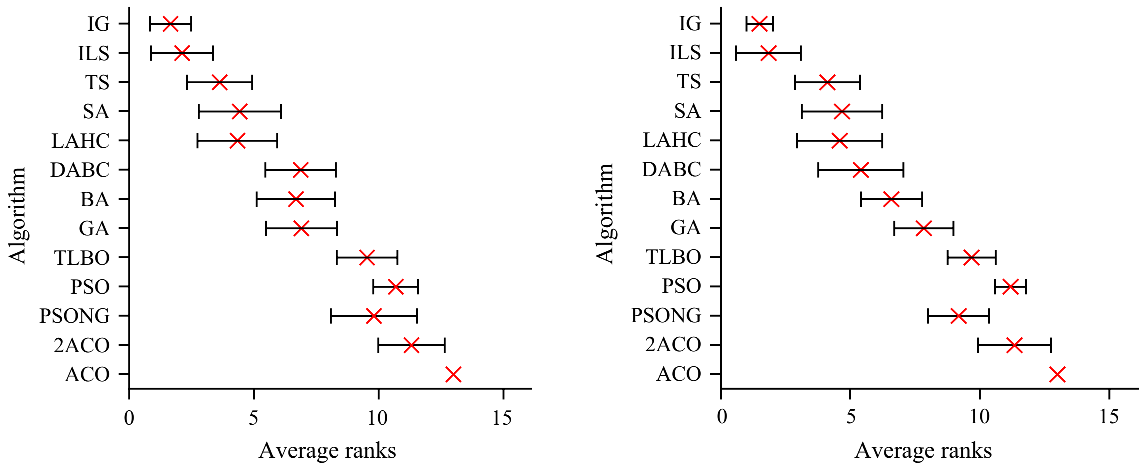
The proper parameters often play an underlying role in creating a high-performing algorithm. Therefore, this section first calibrates the parameters for all the algorithms with full factorial design as proposed by Tang et al. [25] and Li et al. [17]. One of the largest cases from P205-Yuan is used for parameter determination, and this case is solved 10 times for each combination of the parameters. All the algorithms terminate when the computational time reaches $t = Nt \times Nt \times 5$ milliseconds. After carrying out all the experiments, the parametric analysis of variance method (ANOVA) is applied to select the best combination of the parameters. All the final selected parameters, along with the ranges for each parameter, are available upon request. For the MTALBP-I, the workstation number minimization is the primary concern, and thus the following comparison focuses mainly on the results in terms of the number of workstations obtained using station-oriented evaluation.

Table 4 shows the average RDI values for all problems with four termination criteria. It is to be noted that both TS1 and TS4 are utilized, and they are randomly selected for small-sized problems. This is due to that fact TSD4 with reduced research space can lose the optimal solutions for a few small-size cases. STD1, on the other hand, has the largest search space, and the optimal solution is definitely in the search space. All the other decoding methods utilize heuristic methods to decide the side for E-type tasks to reduce search space at the cost of losing optimal solutions. In the preliminary test, this situation occurs only for a few small-sized problems, and thus STD4 is selected exclusively for large-sized problems. In Table 4, the number in each column is the average results for several cycle times with 10 runs. For instance, each number for P205-Yuan is the average result of 100 datasets, combing the results of 10 different cycle times. It is observed that IG is the best algorithm with an overall RDI of 7.3 under $\rho = 5$, and ILS is the second-best algorithm with an overall RDI of 9.9.

For the other three termination criteria, the IG and ILS are also the two best algorithms. Among the remaining methods, the LAHC, SA, and TS, which are also local search methods, outperform the GA, DABC, and BA. Results obtained using the TLBO, PSO, PSONG, and 2ACO are worse, and for all large-sized cases, the ACO reports the worst performance. It is interesting to investigate the cause of these results. Local search methods benefit from the objective function proposed in this research and ST4 with reduced search space. The newly proposed objective guides them quickly to the near optimal solutions by preserving small improvements. The TLBO and PSO lack strong local search capacity, and thus they are outperformed by the GA, DABC, and BA algorithms.

Although the difference between the proposed algorithms and other competing algorithms is quite large, it is still recommended that a statistical analysis be carried out to check whether the observed difference is statistically significant. Since an initial analysis with a parametric ANOVA technique shows strong deviation from normality, a non-parametric Friedman rank-based analysis is also executed. As there are 13 methods, the results are transferred in a way such that the best result is given a rank of 1, and the worst result is given a rank of 13. Because four termination criteria are applied, there are four statistical results with Friedman rank-based analysis. On the basis of the statistical analysis, it is observed that there are statistical differences among these algorithms with P-values less than 0.01 for all four termination conditions. Instead of exhibiting the detailed statistical results, this paper mainly presents the average ranks of the 13 algorithms on large-sized cases in Fig. 7a, b. Figure 7a depicts the average ranks with 95% minimal significant difference confidence intervals when $\rho = 5$, and Fig. 7b depicts the average ranks with confidence intervals when $\rho = 20$, in order to show the performances with the smallest and largest CPU times.

From these figures, it is clear that the IG and ILS rank first and second respectively, for both termination criteria. They are followed by the TS, SA, and LAHC, ranking third, fourth, and fifth. Subsequently, the DABC, BA, and GA rank sixth, seventh, and eighth, and the ACO ranks last. It is also observed that the 95% confidence intervals of the IG and ILS almost do not overlap with the confidence intervals of the



(a) Ranks of the algorithms under $\rho = 5$ **(b)** Ranks of the algorithms under $\rho = 20$

Fig. 7 Means plot of the average ranks and 95% confidence intervals for 13 algorithms

Table 5 New lower bounds and best results by algorithms under $\rho = 20$

Problem	CT	LB-O	LB-N	OPT	HBMO	MPSO	GA	DABC	BA	LAHC	SA	TS	IG	ILS
P9	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	5	3	3	3	3	3	3	3	3	3	3	3	3	3
	6	3	3	3	3	3	3	3	3	3	3	3	3	3
P12	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	6	4	4	4	4	4	4	4	4	4	4	4	4	4
	7	3	**4**	4	4	4	4	4	4	4	4	4	4	4
	8	3	3	3	3	3	3	3	3	3	3	3	3	3
P16	15	5	5	6	6	6	6	6	6	6	6	6	6	6
	16	5	5	6	6	6	6	6	6	6	6	6	6	6
	18	4	4	5	5	5	5	5	5	5	5	5	5	5
	19	4	4	5	5	5	5	5	5	5	5	5	5	5
	21	4	4	4	4	4	4	4	4	4	4	4	4	4
	22	4	4	4	4	4	4	4	4	4	4	4	4	4
P24	20	7	7	7	7	7	7	7	7	7	7	7	7	7
	24	6	6	6	6	6	6	6	6	6	6	6	6	6
	25	5	**6**	6	6	6	6	6	6	6	6	6	6	6
	30	5	5	5	5	5	5	5	5	5	5	5	5	5
	35	4	4	4	4	4	4	4	4	4	4	4	4	4
	40	4	4	4	4	4	4	4	4	4	4	4	4	4
P65	326	8	8	N/A	9	9	9	9	9	9	9	9	9	9
	381	7	7	7	8	8	8	8	8	8	8	7	7	7
	435	6	6	6	7	7	7	7	7	6	6	7	6	6
	490	6	6	6	6	6	6	6	6	6	6	6	6	6
	544	5	5	5	6	6	5	5	5	5	5	5	5	5
P148	204	13	**14**	14	17	17	15	15	15	15	15	15	15	14
	255	11	**12**	12	13	13	12	12	12	12	12	12	12	12
	306	9	**10**	10	11	11	10	10	10	10	10	10	10	10
	357	8	8	8	10	9	8	9	8	8	8	8	8	8
	408	7	7	7	8	8	7	7	7	7	7	7	7	7
	459	6	**7**	7	7	7	7	7	7	7	7	7	7	7
	510	6	6	6	7	6	6	6	6	6	6	6	6	6
P205-Yuan	1133	11	**12**	N/A	15	N/A	13	13	13	13	13	13	13	13
	1322	10	**11**	11	13	N/A	11	11	11	11	11	11	11	11
	1510	8	**9**	N/A	11	N/A	10	10	10	10	10	10	10	10
	1699	8	8	N/A	10	N/A	9	9	9	9	9	9	9	9
	1888	7	**8**	8	9	N/A	8	8	8	8	8	8	8	8
	2077	6	**7**	7	8	N/A	7	7	7	7	7	7	7	7
	2266	6	6	N/A	8	N/A	7	7	7	7	7	7	7	7
	2454	5	**6**	6	7	N/A	6	6	6	6	6	6	6	6
	2643	5	**6**	6	6	N/A	6	6	6	6	6	6	6	6
	2832	5	5	5	6	N/A	5	5	5	5	5	5	5	5

Table 5 (continued)

Problem	CT	LB-O	LB-N	OPT	HBMO	MPSO	GA	DABC	BA	LAHC	SA	TS	IG	ILS
P205-Delice	1133	11	**12**	12	N/A	14	13	13	13	13	13	13	13	12
	1322	9	**10**	N/A	N/A	12	11	11	11	11	11	11	11	11
	1510	8	**9**	9	N/A	11	10	10	10	10	9	9	9	9
	1699	7	**8**	8	N/A	10	8	8	8	8	8	8	8	8
	1888	7	7	N/A	N/A	8	8	8	8	8	8	8	8	8
	2077	6	**7**	7	N/A	8	7	7	7	7	7	7	7	7
	2266	6	6	6	N/A	8	6	6	6	6	6	6	6	6
	2454	5	**6**	6	N/A	6	6	6	6	6	6	6	6	6
	2643	5	5	N/A	N/A	6	6	6	6	6	6	6	6	6
	2832	5	5	5	N/A	6	5	5	5	5	5	5	5	5

*N/A means that the result is not available, and new lower bounds are in bold

other algorithms. This underlines that the proposed IG and ILS are better than other methods by a clear and significant margin.

To the authors' best knowledge, there are several cases in which the optimal solutions are not found, and the gap between the obtained best results and the theoretical lower bounds by Özcan and Toklu [18] are quite large. Özcan and Toklu [18] calculated the lower bounds using the weighted task times and lower bound calculation method as shown in Hu et al. [6]. This method suits the weighted task time situation, but they are not efficient for the non-weighted situation considered in this research. Therefore, this research presents an improved version for lower bound calculation with expressions (15–18). In these expressions, LB_m^{Nm} and LB_m^{Ns} are the lower bounds of the number of mated-stations, and the number of workstations for the model m; LB^{Nm} and LB^{Ns} are the lower bounds of the number of mated-stations and the number of workstations for the whole problem. i indicates a task, t_{im} is the operation time for task i of model m, and AL, AR, and AE are sets of tasks whose preferred directions are left, right, and either, respectively. The logic behind the improvement is that each product needs to be completed within the cycle time for the non-weighted method.

$$LB_m^{Nm} = \max\left\{ \sum_{i\in AL} t_{im}/CT, \sum_{i\in AR} t_{im}/CT \right\} \tag{15}$$

$$LB_m^{Ns} = 2 \cdot LB_m^{Nm} + \max\left\{ 0, \frac{\sum_{i\in AE} t_{im} - \left(LB_m^{Nm} \cdot CT - \sum_{i\in AL} t_{im}\right) - \left(LB_m^{Nm} \cdot CT - \sum_{i\in AR} t_{im}\right)}{CT} \right\} \tag{16}$$

$$LB^{Nm} = \max_m \left\{ LB_m^{Nm} \right\} \tag{17}$$

$$LB^{Ns} = \max_m \left\{ LB_m^{Ns} \right\} \tag{18}$$

Table 5 presents the lower bounds and best workstation numbers by part of the algorithms using the largest computational time within 10 repetitions. The old lower bounds (LB-O) provided by Özcan and Toklu [18] are highlighted by adding an "–O" at the end of the name, and the new lower bounds (LB-N) proposed in this paper are highlighted by adding an "–N." OPT means the optimal number of workstations. For small-sized problems, the optimal results have been provided. For large-sized problems, the solution is optimal when the number of workstations is equal to the lower bound of the workstation number. The current best results obtained by modified particle swarm optimization (MPSO) [5] and hybrid honey bee mating optimization (HBMO) [27] are also included.

From Table 5, it can be seen that the LB-N updates the lower bounds for 19 cases, including two small-sized cases and 17 large-sized cases. Using this new lower bound calculation, the results for 24 large-sized cases are proven to be optimal, among which 14 cases can be proved to be optimal only through the new lower bound calculation. Also, all eight of the proposed algorithms in this paper can obtain the same or better results when compared with MPSO and HBMO for all large-sized cases. To be specific, the IG outperforms HBMO in three, six, and nine cases for P65, P148, and P205-Yuan, respectively. Moreover, the IG outperforms MPSO in three, five, and seven cases for P65, P148, and P205-Delice, respectively. Among all the methods, the IG and ILS find the maximum number of optimal solutions. Among the large-sized problems, 19 optimal results are first presented using the IG and ILS. These results further confirm the advantages of the proposed local search methods as well as the new station-oriented evaluation.

7 Conclusion and future research

Mixed-model two-sided assembly lines are utilized in modern industry to assemble different types of products simultaneously and in parallel, and this type of line has great relevance to the automobile industry. This paper considers this practical and important mixed-model two-sided assembly line balancing problem to minimize the mated-station number and workstation number (MTALBP-I).

Firstly, six new decoding schemes are developed for the MTALBP-I (modified from those used for solving the TALBP) for the first time, and a comparative study of nine decoding procedures is also carried out to test their performance. Secondly, two local search methods, the iterated greedy (IG) and iterated local search (ILS), are developed to solve the MTALBP-I. A new precedence-based local search using referenced permutation and two neighborhood structures is also employed by these two methods to emphasize intensification while preserving fast search speed. A two-stage evaluation procedure is developed to guide the search process, wherein the first stage with a station-oriented evaluation is applied to find the solutions with the fewest workstations, and the second stage attempts to find balanced solutions on the basis of the solution obtained in the first stage. Thirdly, a comprehensive study of these two algorithms and 11 adaptions of recent and high-performing algorithms demonstrates that the proposed algorithm outperforms other implemented methods and obtains 23 new upper bounds. Fourthly, a new lower bound calculation is developed, which updates the lower bounds for 19 problem cases and proves the optimality for 14 more cases.

The developed techniques can be adopted by the production-line managers to reduce the number of workers, making their assembly lines shorter, improve line efficiency, and thus reduce costs and make factories more competitive. As the real industrial contexts are diverse and complicated, the considered problem can be extended to consider more realistic objectives (e.g., minimizing the cost or the cycle time), the realistic constraints (e.g., positional constraint, zoning constraints), and the uncertainties in production (e.g., uncertain operation times). One can also study the rebalancing of the MTALBP-I so that the existing assembly lines can be improved by utilizing the developed methodologies. As model sequencing is another problem in mixed-model production, it is necessary to study the mixed-model two-sided assembly line balancing and sequencing problem simultaneously.

Acknowledgments This project is partially supported by National Natural Science Foundation of China under Grants 51875421 and 61803287 and the China Postdoctoral Science Foundation under Grant 2018M642928. The authors are grateful for the insightful comments by the anonymous referees which helped to improve this paper.

Appendix A: Illustrated decoding procedure

Utilized notations for decoding procedure

i, h, p	The index of tasks
I	Set of tasks; $i, h \in I$
j	The index of mated-stations
J	Set of mated-stations; $j \in J$
k	The index of sides, $k = 1, 2$
m	The index of the product models
M	Set of product models; $m \in M$
A_L	Set of tasks that should be allocated to the left side of a mated-station
A_R	Set of tasks that should be allocated to the right side of a mated-station
A_E	Set of tasks that should be allocated to either side of a mated-station
$P(h)$	Set of immediate predecessors of task h
t_h^m	Operation time of task h for model m
tf_h^m	Completion time of task h for model m
wl_j^m	The completion time of the left-side workstation of the mated-station j (including the idle time) for model m
wr_j^m	The completion time of the right-side workstation of the mated-station j (including the idle time) for model m
SL_j	Set of tasks that have been allocated to the left side of mated-station j
SR_j	Set of tasks that have been allocated to the right side of mated-station j
ATL_j	Set of assignable tasks that can be allocated to the left side of mated-station j
ATR_j	Set of assignable tasks that can be allocated to the right side of mated-station j
CT	Cycle time
Nt	Total number of tasks
nm, nl, nr	The number of mated-stations, left-side workstation, and right-side workstations
ns	The total number of workstations

The decoding procedure of *TSD4* is detailed as follows, and serves as an example.

```
Step 1: Set $i = 1$, $j = 0$, $nm = 0$, $nl = 0$, $nr = 0$, $wl_j^m = 0\ \forall m \in M$, and $wr_j^m = 0\ \forall m \in M$.
Step 2: If all tasks have been allocated ($i > Nt$), go to Step 7. Otherwise, go to Step 3.
Step 3: Open a new mated-station and set $j = j + 1$, $wl_j^m = 0\ \forall m \in M$, and $wr_j^m = 0\ \forall m \in M$. Afterward, go to Step 4.
Step 4: %Obtain the assignable task set
  For any unallocated task $h$ whose immediate predecessors have been allocated
  4.1 If $h \in A_L$, then
  If $t_h^m + wl_j^m \le CT\ \forall m \in M$ and $t_h^m + tf_r^m \le CT\ \forall m \in M$ ($tf_r^m = max\{tf_p^m | p \in P(h) \cap SR_j\}$), add task $h$ to the assignable task set on the left side $ATL_j$;
  4.2 If $h \in A_R$, then
  If $t_h^m + wr_j^m \le CT\ \forall m \in M$ and $t_h^m + tf_l^m \le CT\ \forall m \in M$ ($tf_l^m = max\{tf_p^m | p \in P(h) \cap SL_j\}$), add task $h$ to the assignable task set on the right side $ATR_j$;
  4.3 If $h \in A_E$, then
  4.3.1 If $t_h^m + wl_j^m \le CT\ \forall m \in M$ and $t_h^m + tf_r^m \le CT\ \forall m \in M$ ($tf_r^m = max\{tf_p^m | p \in P(h) \cap SR_j\}$), task $h$ can be allocated to the left side;
  4.3.2 If $t_h^m + wr_j^m \le CT\ \forall m \in M$ and $t_h^m + tf_l^m \le CT\ \forall m \in M$ ($tf_l^m = max\{tf_p^m | p \in P(h) \cap SL_j\}$), task $h$ can be allocated to the right side;
  4.3.3 If task $h$ can be allocated only to the left (right) side, add this task to the assignable task $ATL_j$ ($ATR_j$) and go to Step 5. If this task can be allocated to either side and $\max_m(wl_j^m) \le \max_m(wr_j^m)$, allocate task $h$ to the assignable task $ATL_j$; otherwise, allocated task $h$ to the assignable task $ATR_j$.
Step 5: If no assignable task exists, set $nm = nm + 1$, $nl = nl + 1$ when $wl_j^m > 0\ \exists m \in M$, $nr = nr + 1$ when $wr_j^m > 0\ \exists m \in M$ and go to Step 2. Otherwise, go to Step 6.
Step 6: % Allocate this task to the selected side.
  Step 6.1: Select an assignable task $h$ in the former position in the task permutation.
  Step 6.2: If task $h$ is in the assignable task $ATL_j$, allocate task $h$ to the left side and set $tf_h^m = max\{t_h^m + wl_j^m, t_h^m + tf_r^m\}$ and $wl_j^m = tf_h^m$. Otherwise, allocate task $h$ to the right side and set $tf_h^m = max\{t_h^m + wr_j^m, t_h^m + tf_l^m\}$ and $wr_j^m = tf_h^m$. Afterward, update $i = i + 1$ and go to Step 4.
Step 7: %Adjust the task assignments on the final mated-station;
  If $\sum_{h \in SL_{nm} \cup SR_{nm}} t_h^m \le CT\ \forall m \in M$, then
  7.1 If $h \in A_L\ \exists h \in SL_{nm}$ and $h \in A_E\ \forall h \in SR_{nm}$, assign all tasks to the left-side workstation and set $nl = nl - 1$;
  7.2 If $h \in A_R\ \exists h \in SR_{nm}$ and $h \in A_E\ \forall h \in SL_{nm}$, assign all tasks to the right-side workstation and set $nr = nr - 1$;
  7.3 $h \in A_E\ \forall h \in SR_{nm} \cup SL_{nm}$, assign all the tasks to the left side workstation and set $nl = nl - 1$.
Output: $nm, nl, nr$ and $ns$
```

References

1. Aghajani M, Ghodsi R, Javadi B (2014) Balancing of robotic mixed-model two-sided assembly line with robot setup times. Int J Adv Manuf Technol 74:1005–1016
2. Bartholdi J (1993) Balancing two-sided assembly lines: a case study. Int J Prod Res 31:2447–2461
3. Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solution approaches. Int J Prod Econ 142:259–277
4. Baykasoglu A, Dereli T (2008) Two-sided assembly line balancing using an ant-colony-based heuristic. Int J Adv Manuf Technol 36:582–588. https://doi.org/10.1007/s00170-006-0861-3
5. Delice Y, Kızılkaya Aydoğan E, Özcan U, İlkay MS (2017) A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. J Intell Manuf 28:23–36. https://doi.org/10.1007/s10845-014-0959-7
6. Hu X, Wu E, Jin Y (2008) A station-oriented enumerative algorithm for two-sided assembly line balancing. Eur J Oper Res 186:435–440. https://doi.org/10.1016/j.ejor.2007.01.022
7. Khorasanian D, Hejazi SR, Moslehi G (2013) Two-sided assembly line balancing considering the relationships between tasks. Comput Ind Eng 66:1096–1105. https://doi.org/10.1016/j.cie.2013.08.006
8. Kim YK, Kim Y, Kim YJ (2000) Two-sided assembly line balancing: a genetic algorithm approach. Prod. Plan. Control 11:44–53. https://doi.org/10.1080/095372800232478
9. Kim YK, Song WS, Kim JH (2009) A mathematical model and a genetic algorithm for two-sided assembly line balancing. Comput Oper Res 36:853–865. https://doi.org/10.1016/j.cor.2007.11.003
10. Kucukkoc I, Li Z, Karaoglan AD, Zhang DZ (2018) Balancing of mixed-model two-sided assembly lines with underground workstations: a mathematical model and ant colony optimization algorithm. Int J Prod Econ 205:228–243. https://doi.org/10.1016/j.ijpe.2018.08.009
11. Kucukkoc I, Zhang DZ (2014) Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. Int J Prod Econ 158:314–333. https://doi.org/10.1016/j.ijpe.2014.08.010
12. Kucukkoc I, Zhang DZ (2016) Mixed-model parallel two-sided assembly line balancing problem: a flexible agent-based ant colony optimization approach. Comput Ind Eng 97:58–72. https://doi.org/10.1016/j.cie.2016.04.001
13. Lee TO, Kim Y, Kim YK (2001) Two-sided assembly line balancing to maximize work relatedness and slackness. Comput Ind Eng 40:273–292. https://doi.org/10.1016/S0360-8352(01)00029-8
14. Li D, Zhang C, Tian G, Shao X, Li Z (2018) Multiobjective program and hybrid imperialist competitive algorithm for the mixed-model two-sided assembly lines subject to multiple constraints. IEEE Trans. Syst. Man Cybern. Syst. 48:119–129. https://doi.org/10.1109/TSMC.2016.2598685
15. Li Z, Kucukkoc I, Nilakantan JM (2017) Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. Comput Oper Res 84:146–161. https://doi.org/10.1016/j.cor.2017.03.002
16. Li Z, Tang Q, Zhang L (2016) Minimizing the cycle time in two-sided assembly lines with assignment restrictions: improvements and a simple algorithm. Math. Probl. Eng. 2016:1–15. https://doi.org/10.1155/2016/4536426
17. Li Z, Tang Q, Zhang L (2017) Two-sided assembly line balancing problem of type I: improvements, a simple algorithm and a comprehensive study. Comput Oper Res 79:78–93. https://doi.org/10.1016/j.cor.2016.10.006
18. Özcan U, Toklu B (2009) Balancing of mixed-model two-sided assembly lines. Comput Ind Eng 57:217–227. https://doi.org/10.1016/j.cie.2008.11.012
19. Özcan U, Toklu B (2009) A tabu search algorithm for two-sided assembly line balancing. Int J Adv Manuf Technol 43:822–829
20. Özcan U, Toklu B (2010) Balancing two-sided assembly lines with sequence-dependent setup times. Int J Prod Res 48:5363–5383. https://doi.org/10.1080/00207540903140750
21. Pan Q-K, Ruiz R (2012) Local search methods for the flowshop scheduling problem with flowtime minimization. Eur J Oper Res 222:31–43. https://doi.org/10.1016/j.ejor.2012.04.034
22. Purnomo HD, Wee H-M, Rau H (2013) Two-sided assembly lines balancing with assignment restrictions. Math Comput Model 57:189–199. https://doi.org/10.1016/j.mcm.2011.06.010
23. Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 177:2033–2049
24. Simaria AS, Vilarinho PM (2009) 2-ANTBAL: an ant colony optimisation algorithm for balancing two-sided assembly lines. Comput Ind Eng 56:489–506
25. Tang Q, Li Z, Zhang L (2016) An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-II. Comput Ind Eng 97:146–156. https://doi.org/10.1016/j.cie.2016.05.004
26. Yang W, Cheng W (2019) Modelling and solving mixed-model two-sided assembly line balancing problem with sequence-dependent setup time. J. Prod. Res, Int. https://doi.org/10.1080/00207543.2019.1683255
27. Yuan B, Zhang C, Shao X, Jiang Z (2015) An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. Comput Oper Res 53:32–41. https://doi.org/10.1016/j.cor.2014.07.011