Front. Mech. Eng. 2019, 14(2): 241–253
https://doi.org/10.1007/s11465-018-0518-6

RESEARCH ARTICLE

Xiaokun DUAN, Bo WU, Youmin HU, Jie LIU, Jing XIONG

An improved artificial bee colony algorithm with MaxTF heuristic rule for two-sided assembly line balancing problem

Abstract Two-sided assembly line is usually used for the assembly of large products such as cars, buses, and trucks. With the development of technical progress, the assembly line needs to be reconfigured and the cycle time of the line should be optimized to satisfy the new assembly process. Two-sided assembly line balancing with the objective of minimizing the cycle time is called TALBP-2. This paper proposes an improved artificial bee colony (IABC) algorithm with the MaxTF heuristic rule. In the heuristic initialization process, the MaxTF rule defines a new task's priority weight. On the basis of priority weight, the assignment of tasks is reasonable and the quality of an initial solution is high. In the IABC algorithm, two neighborhood strategies are embedded to balance the exploitation and exploration abilities of the algorithm. The employed bees and onlooker bees produce neighboring solutions in different promising regions to accelerate the convergence rate. Furthermore, a well-designed random strategy of scout bees is developed to escape local optima. The experimental results demonstrate that the proposed MaxTF rule performs better than other heuristic rules, as it can find the best solution for all the 10 test cases. A comparison of the IABC algorithm and other algorithms proves the effectiveness of the proposed IABC algorithm. The results also denote that the IABC algorithm is efficient and stable in minimizing the cycle time for the TALBP-2, and it can find 20 new best solutions among 25 large-sized problem cases.

Received July 24, 2017; accepted January 30, 2018

Xiaokun DUAN, Bo WU, Youmin HU (✉), Jie LIU
School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
E-mail: youmhwh@hust.edu.cn

Jing XIONG
School of Mechanical Engineering, Hubei Engineering University, Xiaogan 432000, China

Keywords two-sided assembly line balancing problem, artificial bee colony algorithm, heuristic rules, time boundary

1 Introduction

The assembly line balancing problem (ALBP) refers to distributing assembly work among the workstations with respect to a certain objective to reach the desired level of performance. Two kinds of problem versions arise from varying the objective of ALBP. ALBP-1 minimizes the number of stations m given a fixed cycle time c, whereas ALBP-2 minimizes c given m [1].

The assembly lines can be classified into two categories: One-sided and two-sided assembly line, and the main difference between them is that two-sided assembly line owns pairs of stations that are located opposite to each other on the two sides of the assembly line [2]. By contrast, stations are distributed at only one side in one-sided assembly lines. Although many algorithms have been used to solve the one-sided ALBP, such as exact methods [3–5] and heuristic methods [6–8], the applications of exact methods heuristic methods to two-sided ALBP (TALBP) are relatively small. Exact methods are suitable for solving the exact solutions of small-sized problems, but they are inefficient for solving large-sized problems. Heuristic methods are highly efficient for solving one-sided ALBP, but the effect is difficult to guarantee.

Compared with one-sided assembly line, two-sided assembly line has the following advantages: Shorter line length, reduced material handling, reduced throughput time and set-up time, and lower cost of tools/fixtures. Given these advantages, the assembly of large high-volume products, e.g., automobiles, buses, and trucks, are usually organized in two-sided assembly lines. Similar to the ALBP, the TALBP is also a non-deterministic polynomial hard (NP-hard) problem whose complexity grows very fast with the problem size [9]. To design two-

sided assembly lines, some complex restrictions should be considered, such as the direction constraint, precedence constraint, and cycle time constraint of the workstation.

When the two-sided assembly line is installed, the cycle time should be adjusted following different requirements. With the development of technological progress, the task operation time is changed due to the utilization of new equipment or new assembly process, causing the reconfiguration of the assembly line. In this case, the two-sized assembly line needs to be reconfigured and the original cycle time needs to be adjusted to satisfy the new manufacture demands. Therefore, the TALBP-2 is required for further research.

The TALBP is first presented and studied by Bartholdi [10] with a first fit assignment rule implemented by an interactive program. Kim et al. [11] developed a new genetic algorithm to balance the operation time of the TALBP. Wu et al. [12] investigated a branch-and-bound algorithm for the TALBP-1 with the objective of minimizing the length of the line. Baykasoglu and Dereli [13] proposed an ant colony heuristic algorithm to solve the TALBP-1 with the objective of maximizing work relatedness. Özcan and Toklu [14] formulated the mixed model of the TALBP-1 as a mixed integer program and solved it with a simulated annealing algorithm. Kim et al. [15] presented a mathematical model and a genetic algorithm for the TALBP-2. Özcan [16] proposed a chance-constrained, piecewise-linear, mixed integer program for solving the TALBP-1 with stochastic task times. Tapkan et al. [17] investigated the bee algorithm to handle large instances of the TALBP-1. Khorasanian et al. [18] applied a simulated annealing algorithm for solving the TALBP-1 with task consistency. Yuan et al. [19] constructed an integer programming model for the TALBP-1 with constraints and solved it with the late acceptance hill-climbing algorithm. Tang et al. [20] proposed an improved algorithm with idle time reduction techniques to solve the TALBP-2. Li et al. [21] provided an iterated greedy algorithm to address the TALBP-2. Tang et al. [22] proposed a hybrid teaching-learning-based optimization algorithm to balancing the stochastic two-sided assembly line problem. Li et al. [23] presented a comprehensive review, and different meta-heuristics were compared to solve the TALBP-1. Gansterer and Hartl [24] considered the TALBP-1 with real-world constraints and proposed a genetic algorithm to solve the problem. Although many studies have been conducted on TALBP, the TALBP-2 has received limited attention.

Artificial bee colony (ABC) algorithm was proposed by Karaboga [25]. It has been successfully applied to solve multimodal and multidimensional optimization problems, such as image stegnalysis problems [26], scheduling problems [27], and function optimization problems [28]. For the TALBP-2, the decreasing cycle time demands the search precision to be good enough to find a feasible solution quickly. The employed bees and onlooker bees of improved ABC (IABC) can enhance the search precision and accelerate the search speed. The current best solutions may be obstacles in obtaining a new cycle time. The scout bees of IABC can escape local optima by importing new high-performing solutions.

The rest of this paper is structured as follows. Section 2 describes the mathematical formulation of the TALBP. The original ABC algorithm is described in Section 3. Section 4 introduces the proposed heuristic rules and IABC algorithm for the TALBP-2 in detail. Section 5 presents experimental results for five benchmark instances. Finally, Section 6 provides the conclusion and future directions for related research.

2 Problem description

2.1 Problem definition

The two-sided assembly line has two sides in parallel to simultaneously assemble different tasks of the same product, as shown in Fig. 1. Two directly facing stations, such as Stations 1 and 2, are called a mated-station.

The task of the TALBP has a different operation direction compared with that of the one-sided assembly line balancing (ALB). In the TALBP, some tasks must be assigned strictly to the left side (L) or the right side (R), and some tasks can be performed at either side (E) of the line. For example, in a loader assembly line, such tasks as barometer and oil-water separator must be assigned to the left side of the line because these tasks are L-type tasks, whereas the booster pump and throttle must be allocated to the right side as these tasks are R-type tasks. E-type tasks, such as engine and multiple valve, can be assembled to the left or right side of the line.

To solve the TALBP, three constraints must be satisfied:

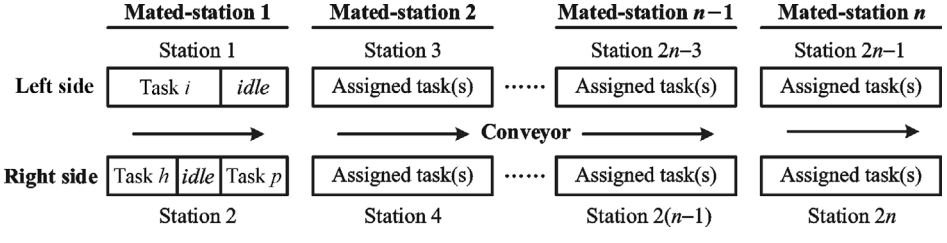

Fig. 1 Two-sided assembly line

The assignment constraints (each task must be assigned to exactly one station), cycle time constraint (the total task times of all the tasks assigned to a station cannot exceed the cycle time), and precedence constraint (no task is assigned to an earlier position than any of its predecessors) [29].

2.2 Mathematical model

The notations used in this paper are listed in Table 1.

Table 1 Notations in TALBP

Natation	Representation
I	Set of tasks, $I=\{1, 2, \ldots, I, \ldots, n\}$
J	Set of mated-stations, $J=\{1, 2, \ldots, j, \ldots, m\}$
k	Indicator for the side of stations, $k=1$, if the station is left side; $k=2$, otherwise
(j, k)	A station of mated-station j and its operation direction is k
S_{L}	Set of tasks assigned to a left station; $S_{\mathrm{L}} \subset I$
S_{R}	Set of tasks assigned to a right station; $S_{\mathrm{R}} \subset I$
S_{E}	Set of tasks assigned to either station; $S_{\mathrm{E}} \subset I$
P_0	Set of tasks that have no predecessors
$P(i)$	Set of immediate predecessors of Task i
$P_{\mathrm{a}}(i)$	Set of all predecessors of Task i
$S(i)$	Set of immediate successors of Task i
$S_{\mathrm{a}}(i)$	Set of all successors of Task i
$D(i)$	Set of tasks whose operations are opposite to Task i's operation direction $D(i)=\begin{cases} S_{\mathrm{L}} & \text{if } i \in S_{\mathrm{R}} \\ S_{\mathrm{R}} & \text{if } i \in S_{\mathrm{L}} \\ \varnothing & \text{if } i \in S_{\mathrm{E}} \end{cases}$
$K(i)$	Set of all predecessors of Task i $K(i)=\begin{cases} \{1\} & \text{if } i \in S_{\mathrm{R}} \\ \{1\} & \text{if } i \in S_{\mathrm{L}} \\ \{1, 2\} & \text{if } i \in S_{\mathrm{E}} \end{cases}$
t_i	Processing time of Task i
ft_i	Finish time of Task i
CT	Cycle time
x_{ijk}	$x_{ijk}=1$, if Task i is assigned to station (j, k); $x_{ijk}=0$, otherwise
z_{ir}	$z_{ir}=1$, if Task i is assigned earlier than Task r in the same station; $z_{ir}=0$, otherwise
φ	A very large positive number

The objective of the TALBP is to minimize the cycle time for a given number of mated-stations. The mathematical model formulation is as follows:

$$\text{Minimize } CT. \tag{1}$$

Subject to

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

$$\sum_{g \in J} \sum_{k \in K(h)} g x_{hgk} \leqslant \sum_{j \in J} \sum_{k \in K(i)} j x_{igk},\ i \in I - P_0,$$

$$h \in P(i), \tag{3}$$

$$ft_i \leqslant CT,\ i \in I, \tag{4}$$

$$ft_i - ft_h + \varphi\left(1 - \sum_{k \in K(h)} x_{hjk}\right) + \varphi\left(1 - \sum_{k \in K(i)} x_{ijk}\right) \geqslant t_i,$$

$$i \in I - P_0,\ h \in P(i),\ j \in J, \tag{5}$$

$$ft_r - ft_i + \varphi(1 - x_{rjk}) + \varphi(1 - x_{ijk}) + \varphi(1 - z_{ir}) \geqslant t_r,$$

$$i \in I,\ j \in J,\ k \in K(i) \cap K(r),$$

$$r \in \left\{e | e \in I - \left(P_{\mathrm{a}}(i) \cup S_{\mathrm{a}}(i) \cup D(i)\right),\ i < e\right\}, \tag{6}$$

$$ft_i - ft_r + \varphi(1 - x_{rjk}) + \varphi(1 - x_{ijk}) + \varphi z_{ir} \geqslant t_i,$$

$$i \in I,\ j \in J,\ k \in K(i) \cap K(r),$$

$$r \in \left\{e | e \in I - \left(P_{\mathrm{a}}(i) \cup S_{\mathrm{a}}(i) \cup D(i)\right),\ i < e\right\}, \tag{7}$$

$$x_{ij1} = 1,\ i \in S_{\mathrm{L}},\ j \in J, \tag{8}$$

$$x_{ij2} = 1,\ i \in S_{\mathrm{R}},\ j \in J, \tag{9}$$

$$x_{ijk} \in \{0, 1\},\ i \in S_{\mathrm{E}},\ j \in J,\ k = 1, 2, \tag{10}$$

$$z_{ir} \in \{0, 1\},\quad i \in I,$$

$$r \in \left\{e | e \in I - \left(P_{\mathrm{a}}(i) \cup S_{\mathrm{a}}(i) \cup D(i)\right),\ i < e\right\}, \tag{11}$$

$$ft_i \geqslant t_i,\ i \in I. \tag{12}$$

The objective Eq. (1) is to minimize the cycle time. Constraint Eq. (2) is the occurrence constraint, which means every task must be assigned to only one station. Constraint Eq. (3) is the precedence constraint, which ensures any task cannot be assigned before its predecessors are completed. Constraints Eqs. (4)–(7) are cycle time constraints, which ensure that each task is finished in the cycle time and the sequence-dependent finish time of tasks is satisfied [14]. For each pair of Tasks i and h, if Task h is an immediate predecessor of Task i and the two tasks are assigned to the same mated-station j, then constraint Eq. (5) becomes active, i.e., $ft_i \geqslant ft_h + t_i$. If the two Tasks i and r have no precedence constraints and are assigned to the

same station (j, k), then constraints Eqs. (6) and (7) become active. If Task i is assigned earlier than Task p, then constraint Eq. (6) becomes $ft_p \geqslant ft_i + t_m$. Otherwise, constraint Eq. (7) becomes $ft_i \geqslant ft_p + t_i$. Constraints Eqs. (8)–(12) are the integrality constraints.

3 Artificial bee colony algorithm

The ABC algorithm is a relatively new member of swarm intelligence. It is inspired by the definite aggregate behavior of honey bee swarms.

The ABC algorithm contains three types of bees, namely, employed bees, onlooker bees, and scout bees. In general, half of the colony is composed of employed bees and the other half consists of onlookers. In the initial population of the ABC algorithm, each of the randomly generated individuals in the population is called a food source and represented by an n-dimensional real-valued vector, which represents a solution of the problem. Let $X_i=\{x_{i1}, x_{i2}, \ldots, x_{in}\}$ represent the ith food source in the population, and each solution is generated as follows:

$$x_{ij} = LB_j + (UB_j - LB_j)r, \tag{13}$$

where $i = 1, 2, \ldots, SN$, $j = 1, 2, \ldots, n$, r is a uniform random number between [0, 1], and LB_j and UB_j are the lower and upper bounds for the dimension j, respectively.

Each employed bee exploits a new solution Y_i from the neighborhood of a population solution and chooses a better one by the greedy algorithm. The new solution Y_i is generated by the following expression:

$$y_{ij} = x_{ij} + (x_{ij} - x_{kj})\theta, \tag{14}$$

where $k = 1, 2, \ldots, SN$ and $j = 1, 2, \ldots, n$ are randomly chosen indexes, k is different from i, and θ is a uniform random number in the range $[-1, 1]$.

The fitness of each solution will be evaluated and shared with onlooker bees. Each onlooker bee chooses a solution depending on its probability value. The solution with the bigger fitness value has a higher possibility to be selected than that with a smaller fitness value. The probability value p_i of the solution is calculated as follows:

$$p_i = \frac{fit_i}{\sum_{j=1}^{n} fit_j}, \tag{15}$$

where fit_i is the fitness value of the solution X_i.

As in the case of the employed bees, onlooker bees exploit and update the corresponding solution by the greedy algorithm.

If the solution cannot be improved after a certain number of generations, then it will be abandoned and replaced by a random solution using Eq. (13). The random solution is explored by the corresponding bee as a scout bee.

Another iteration of the procedure repeats again until a termination is satisfied. For more details on the ABC algorithm, refer to Karaboga [25].

4 Improved artificial bee colony for TALBP-2

In the original ABC algorithm, the employed bee strategy and onlooker bee strategy can be improved to enhance the exploitation ability. The ABC algorithm demonstrates enhanced convergence ability, and it generally traps in local optima in the scout bee phase. Therefore, this paper proposes an IABC algorithm to solve the TALBP-2.

In the IABC algorithm, the heuristic rules are applied to the initial population to improve the quality of the individuals. Meanwhile, artificial bees are adopted in the neighborhood solution strategy to enhance the exploitation ability of the IABC algorithm. This step differs from the basic ABC algorithm, which adopts a random solution strategy for the artificial bees. The flowchart of the proposed IABC algorithm is shown in Fig. 2.

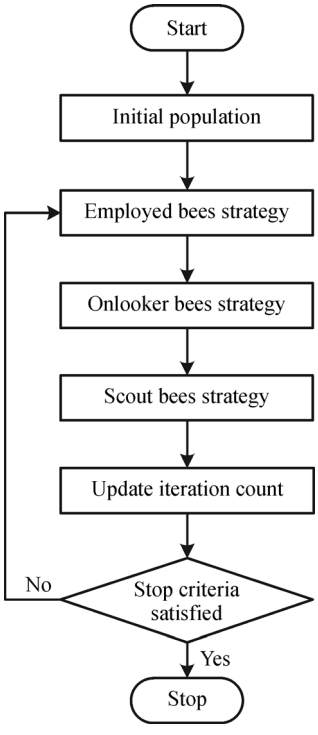

Fig. 2 Flowchart of the IABC algorithm

4.1 Encoding

To solve the TALBP, a permutation-based representation is used. In this paper, each feasible solution is represented by a string of integers, and the length of the string is equal to the number of tasks. Each element consists of a task to be assigned and the operation direction of the task. The locations of elements are the assembly sequences of tasks.

An example of P16 (16-tasks of TALBP) is depicted in Fig. 3, where the number of each task in the circle is labeled with (t_i, d). t_i means the operation time of Task i, and d means the preferred operation direction. The arrow represents the precedence constraints among tasks.

The solution of P16 can be represented as an individual (Fig. 4). It means Task 1 is initially assigned to the left station, followed by Task 4, which is also assigned to the left station, and so on. Finally, Task 15 is assigned to the left station.

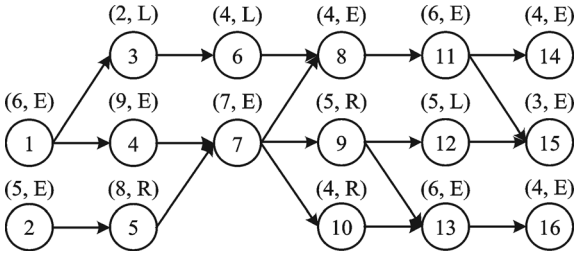

Fig. 3 Example of TALBP: P16

4.2 Decoding

An encoding includes the assembly sequence and operation direction of each task, but it does not indicate the number of mated-stations of a task. Therefore, the assignment scheme of a solution should be interpreted by decoding. The procedure of decoding is as follows:

Step 1: Activate a mated-station, set $j = 1$.

Step 2: Assign Task i to the specified station in accordance with the assembly sequence.

Step 3: Evaluate the start time st_i of Task i. If $st_i + t_i > CT$, (the cycle time CT is determined under the obtained solution), then activate the next mated-station, and set $j = j + 1$, $st_i = 0$. Assign the Task i to the new mated-station.

Step 4: If all the tasks have been decoded, then stop the procedure. Otherwise, go back to Step 2.

In Step 3, the start time is the latest finish time of all the predecessor tasks.

Under the case in Fig. 4 as an example, we assume the cycle time $CT = 17$. The corresponding assignment scheme of ALB can be represented as Fig. 5. In this Gantt chart, the value "1L" means the left station of mated-station 1, and the number at the upper-right corner of Task i means the finish time of the task.

4.3 Heuristic rule

In the procedure of encoding, the sequence of assigning tasks is decided by the tasks' priority values. The task with the highest priority is first assigned. The tasks' priority values can be calculated by various heuristic rules, such as the task time and precedence constraint. Some of the most effective heuristic rules are MaxT rule (the task with the longest operation time has the highest priority) [30], MaxF rule (the task with the most successor tasks has the highest priority) [31], and MaxRPW rule (the task with the maximum ranked positional weight has the highest priority) [32]. This paper develops a new heuristic rule, namely, MaxTF rule. The rule combines heuristic factors of MaxT rule and MaxF rule; hence, the priority value of each task can be calculated by the following expression:

$$V_i = \alpha \frac{t_i}{\sum_{i=1}^{n} t_i} + \beta \frac{N_i}{\sum_{i=1}^{n} N_i}, \tag{16}$$

1L	4L	2R	5R	7R	3L	9R	6L	8L	12L	10R	13R	16R	11L	14L	15L

Fig. 4 Encoding of a solution of P16

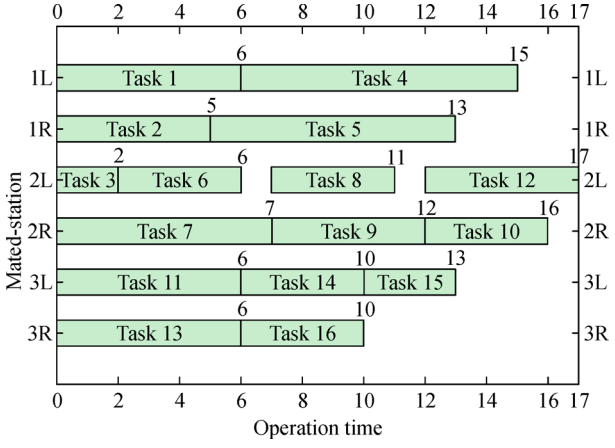

Fig. 5 Gantt chart for a solution of P16

where N_i is the number of successor tasks of the Task i, and α and β is the weight of the heuristic factors ($\alpha + \beta = 1$).

The task assignment rule is described clearly as follows. If a task satisfies all assignment constraints, then it is added into a candidate set. Subsequently, the priority value of these tasks is evaluated. The task with the maximum priority value V_i is selected preferentially in the set. When a tie occurs, the task is selected randomly among the tasks with the same V_i. If the selection task is an E-type task, then it will be assigned to the direction with the earliest start time. In case the earliest start time of both directions is the same, then we assign the task to either left or right direction with uniform probability.

4.4 Population initialization

To ensure the quality and diversity of the individuals in the initial population, the IABC algorithm initializes the population as follows:

Step 1: Activate a mated-station, set $j = 1$ and set $Tms = 0$ (Tms is the operation time of the mated-station). Create a set of assignable tasks: AT.

Step 2: Select a Task i following the heuristic rules, $i \in AT$, and set $Tms = Tms + t_i$.

Step 3: Compare Tms with the time boundary TB. If the operation time $Tms \leqslant \sum_{i=1}^{n} t_i/(2m) + \sum_{i=1}^{n} t_i/n$, assign the Task i to the mated-station j, and record the number and direction of the task to the encoding. The location of the element is the assembling sequence of tasks. Otherwise, set $j = j + 1$ and $Tms = 0$, and assign the task to the new mated-station, set $Tms = Tms + t_i$.

Step 4: If $j = m$, assign all the remaining tasks to the mated-station m, stop the procedure, and obtain a feasible solution. Otherwise, go to Step 5.

Step 5: Update the set AT, and go back to Step 2.

In Step 2, the MaxTF rule and random heuristic rule are adopted in the combination with the proportion of 1:1. In Step 3, the criterion of a new mated-station is $Tms > Tmc$, $Tmc = \sum_{i=1}^{n} t_i/(2m)$, where Tmc means the theoretical minimum cycle time. The operation time of the first $m-1$ mated-stations can be less than TB, but the last mated-station will be seriously overloaded. Therefore, to better control the workload of all the stations and ensure the quality of the solution, the time boundary is revised as $TB = \sum_{i=1}^{n} t_i/(2m) + \sum_{i=1}^{n} t_i/n$.

4.5 Neighborhood strategy

After the initial population procedure, each employed bee and onlooker bee exploits a new solution in the neighborhood of a current solution. In this paper, the new solution is guaranteed not to violate the precedence constraints by adopting two-point crossover and insert mutation as neighborhood strategy to obtain a new solution.

The procedure of the two-point crossover is as follows:

Step 1: Divide parents P1 and P2 into head, middle, and tail portions by two different random points from [1, n].

Step 2: The head and tail portions of offspring O1 are copied from the elements at the same locations in parent P1.

Step 3: The remaining elements of offspring O1 are taken from the elements of parent P1 in the order of elements as in parent P2, as shown in Fig. 6.

Step 4: A similar procedure is operated to offspring O2.

The procedure of insert mutation is as follows:

Step 1: Determine the mutation solution on the basis of the individual mutation rate p_m.

Step 2: Find all immediate predecessor and immediate successor tasks of the mutation elements m. Record the location $m1$ whose immediate predecessor task is the latest assigned, and the location $m2$ whose immediate successor task is the earliest assigned.

Step 3: Select a random integer from ($m1$, $m2$) and insert the mutation element into the location.

4.6 Employed bee strategy

In the IABC algorithm, employed bees explores new neighborhood solutions using the two-point crossover strategy combined with the insert mutation strategy to enhance the exploration ability. The procedure is detailed as follows:

Step 1: Randomly select a current solution $Sc(i)$ in the population.

Step 2: Explore a neighborhood solution from the current solution, as discussed in Section 4.5. Another parent is selected from the initial population. Evaluate the value of CT for these solutions, and select the best one as the new solution $Sn(i)$.

Step 3: Compare CT of $Sc(i)$ and $Sn(i)$. Update the current solution by the greedy algorithm.

Step 4: Compare CT of the best solution $Sbest$ until now and $Sn(i)$. Update the best solution by the greedy algorithm.

4.7 Onlooker bee strategy

To improve the local search ability of the IABC algorithm, onlooker bees further exploit the current population, which is selected by the tournament strategy on the basis of the selection probability value. In accordance with the 10 new neighborhood solutions exploited by onlooker bees, the best and worst solutions are refreshed. The detailed steps are executed as follows:

Step 1: Select a current solution from the new population using the tournament strategy, and the size is set to 3.

Step 2: Exploit 10 neighborhood solutions from the current solution with the two-point crossover strategy. Another parent is the best solution. Evaluate the value of

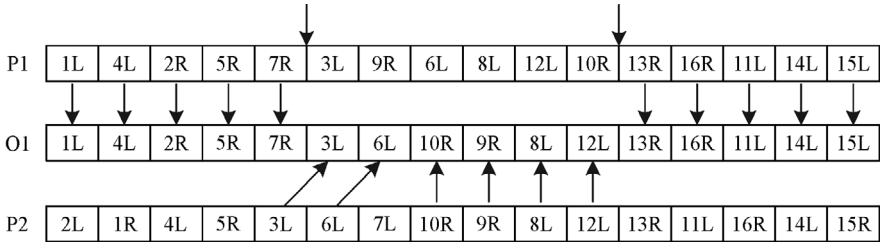

Fig. 6 Example for the two-point crossover

CT for these solutions, and select the best one as the new solution *Sn*(*i*).

Step 3: Compare *CT* of *Sc*(*i*) and *Sn*(*i*). Subsequently, update the current solution by the greedy algorithm.

Step 4: Update the worst solution *Sworst* and the best solution *Sbest* using the greedy algorithm.

4.8 Scout bee strategy

When a solution cannot be improved during limit generation, the solution will be abandoned and replaced and the employed bee will become a scout. To find a new high-quality solution by the scout bee and escape from the local optima, a set of new solutions will be generated randomly and then the best one is selected. If the new solution is better than *Sbest*, then *Sbest* is updated.

5 Experimental results

To evaluate the performance of MaxTF rule and the IABC algorithm for the TALBP-2, computational experiments are applied to five benchmark problems with different numbers of mated-stations. These problems are P24, P65, P148 [11], P16, and P205 [15], where P16 and P24 are small-sized problems, and P65, P148, and P205 are large-sized problems in practical applications. The IABC algorithm is programmed by MATLAB 7.1 on a computer with Intel Core i7 3.4 GHz, 8 GB RAM. To investigate the stability of the proposed algorithm, each compared algorithm is repeated 20 times for all experimental problems. The performance measure of the algorithm is relative percentage deviation (*RPD*). The *RPD* is calculated by

$$RPD = \frac{Some_{\text{sol}} - Best}{Best} \times 100\%, \tag{17}$$

where $Some_{\text{sol}}$ is one solution collected by a given compared algorithm, and *Best* is the best solution obtained among these algorithms.

5.1 Effectiveness of the proposed heuristic rules

A series of tests for the parameters is conducted to analyze the influences of parameters α and β for the performance of the proposed heuristic rules. The minimum cycle times *Tmc* of the best, mean, and standard deviation of the best solutions are reported in Table 2.

The efficiency of the MaxTF rule is the best when the ratio of α and β is 2:8. The best solution and mean value (in bold) at this ratio by heuristic rules are much bigger than others.

To demonstrate the performance of the MaxTF rule, an experiment is performed to compare the efficiency of three other heuristic rules, as shown in Table 3. The last three columns of Table 3 are improved rates of the MaxTF rule compared with MaxT, MaxF, and MaxRPW rules. The improved rate is computed by 100×(Mean of one heuristic rule − Mean of MaxTF)/(Mean of one heuristic rule). Table 3 shows that the proposed MaxTF rule obtains all the best solutions, and the improved rate demonstrates that the MaxTF rule outperforms the other heuristic rules. The MaxTF heuristic rule can improve the quality of the solutions in population initialization.

5.2 Computational results of IABC and other algorithms

In the proposed IABC algorithm, the parameters are set as follows: The size of initial population is set to $Np = 50$ on the basis of preliminary experiments; the numbers of employed bees, onlooker bees, and scout bees are set to $Np/2$, $Np/2$, and 1, respectively; the number of neighborhood solutions is set to $Nns = 10$; the limited number of cycles through which a solution can be passed without improvement is set to $limit = 20$; the stop condition is when the computational times reaches $Ts = n \times n \times 15$ ms, where n is the number of tasks.

For small-sized problems, the optimal solutions are compared with mixed-integer programming (MIP), which is a discretized version of the basic ABC algorithm (DABC) and IABC algorithms. The optimal solutions of MIP are solved by CPLEX Version 12.6.

As shown in Table 4, all the algorithms can find the optimal solution of five small cases. For P24, MIP finds that the computational time of optimal solutions exceeds 3600 s, which is marked with an asterisk. The superiority of IABC is proven by its lower computational time compared with MIP.

Given that MIP cannot find the optimal solution for large-sized problems within an acceptable computational

Table 2 Comparison of the parameter values for the MaxTF rule

Problem	*m*	*Tmc*	α:β = 2:8			α:β = 3:7			α:β = 4:6			α:β = 5:5			α:β = 6:4			α:β = 7:3			α:β = 8:2		
			Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
P16	2	20.50	26	27.2	0.63	**22**	**25.4**	2.46	24	26.0	1.89	**22**	25.9	1.91	**22**	29.2	3.43	27	28.7	1.77	26	28.6	2.46
	3	13.67	**16**	16.9	0.74	**16**	17.0	0.67	**16**	17.2	0.79	**16**	**16.7**	0.67	**16**	16.8	0.42	21	27.6	4.03	21	26.2	3.36
P24	2	35.00	**36**	**38.1**	2.84	37	43.0	3.92	39	43.9	2.81	45	48.3	5.76	44	48.3	5.14	44	50.1	3.57	49	52.8	2.86
	3	23.33	**25**	**26.9**	2.23	**25**	30.7	3.34	30	32.2	4.94	32	41.9	5.90	30	34.2	3.26	30	37.2	2.57	30	37.4	2.67
	4	17.50	**18**	**19.7**	2.06	20	24.5	2.92	19	24.4	2.76	26	29.4	4.99	25	27.2	2.78	25	28.8	2.62	26	33.0	6.55
P65	4	637.4	655	**659.3**	2.62	**649**	659.4	12.51	657	674.7	14.74	662	685.8	16.88	740	852.7	96.74	736	826.4	93.97	706	723.8	19.51
	5	509.9	**532**	**536.1**	3.98	540	559.1	18.28	536	599.1	57.37	558	584.6	36.97	695	769.4	79.80	573	691.2	73.21	549	597.3	62.73
	6	424.9	450	**452.7**	5.01	457	490.8	62.95	**445**	490.8	67.35	452	480.5	38.18	670	720.3	53.54	469	586.9	92.63	456	499.7	76.60
	7	364.2	**382**	404.2	14.58	410	439.1	14.81	385	**393.1**	7.45	456	507.2	46.97	454	572.2	81.10	392	490.1	79.28	399	420.8	37.60
	8	318.7	**336**	**364.5**	18.34	346	417.1	51.58	343	414.8	72.79	340	417.4	99.69	360	488.5	91.58	357	450.1	50.72	353	407.6	62.59

Note: Std = Standard deviation

Table 3 Comparison of the performance for the MaxTF rule

Problem	*m*	*Tmc*	MaxT			MaxF			MaxRPW			MaxTF			Improved rate 1/%	Improved rate 2/%	Improved rate 3/%
			Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std			
P16	2	20.50	33	33.7	0.95	29	29.0	0.00	29	29.0	0.00	**26**	**27.2**	0.63	19.29	6.21	6.21
	3	13.67	33	33.8	1.23	29	29.0	0.00	29	29.0	0.00	**16**	**16.9**	0.74	50.00	41.72	41.72
P24	2	35.00	44	53.4	4.95	**36**	38.7	2.31	40	41.8	2.90	**36**	**38.1**	2.84	28.65	1.55	8.85
	3	23.33	34	35.9	1.20	**25**	27.9	3.31	26	28.3	3.53	**25**	**26.9**	2.23	25.07	3.58	4.95
	4	17.50	19	23.8	3.49	19	23.9	1.91	22	25.4	2.80	**18**	**19.7**	2.06	17.23	17.57	22.44
P65	4	637.4	954	992.5	67.15	725	754.3	26.62	746	783.7	25.39	**655**	**659.3**	2.62	33.57	12.59	15.87
	5	509.9	737	897.6	80.28	653	699.5	27.45	701	716.1	26.33	**532**	**536.1**	3.98	40.27	23.36	25.14
	6	424.9	655	725.7	75.30	562	613.7	28.37	561	590.4	22.85	**450**	**452.7**	5.01	37.62	26.23	23.32
	7	364.2	571	668.8	49.10	513	549.3	30.41	483	538.8	25.15	**382**	**404.2**	14.58	39.56	26.42	24.98
	8	318.7	502	595.4	97.28	463	481.5	17.58	459	474.3	16.32	**336**	**364.5**	18.34	38.78	24.30	23.15

Note: Std = Standard deviation

Table 4 Comparison of the performance for small-sized problems

Problem	*Nms*	LB	MIP			DABC			IABC		
			Best	Mean	Time/s	Best	Mean	Time/s	Best	Mean	Time/s
P16	2	20.50	**22**	**22**	0.16	**22**	**22**	3.84	**22**	**22**	3.84
	3	13.67	**16**	**16**	0.97	**16**	**16**	3.84	**16**	**16**	3.84
P24	2	35.00	**35**	**35**	*	**35**	**35**	8.64	**35**	**35**	8.64
	3	23.33	**24**	**24**	*	**24**	**24**	8.64	**24**	**24**	8.64
	4	17.50	**18**	**18**	*	**18**	**18**	8.64	**18**	**18**	8.64

Note: LB = Lower bound

time, the computational results for large-sized problems are compared with FFR, DABC, *n*-GA, and IABC. For each algorithm, the best, mean, and standard deviation of the best solutions (in bold) are presented in Table 5.

As shown in Table 5, the IABC algorithm obtains 22 best solutions, including 20 brand new best solutions, among 25 large-sized problem cases. The computational results demonstrate that IABC obtains better solutions than FFR and DABC for all the problems, and the experimental values are close to the theoretical optima. However, the IABC algorithm does not surpass *n*-GA for all problems. The IABC algorithm has a slightly worse solution and three negative improved rates for P148 with four and five mated-stations and P205 with four mated-stations.

Table 5 Comparison of the performance for large-sized problems

Problem	*Nms*	LB	FFR			DABC			*n*-GA			IABC			Time/s
			Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	
P65	4	637.4	661	665.7	1.60	645	637.4	2.69	641	643.7	0.39	**640**	**641.4**	0.34	63.375
	5	509.9	528	531.9	1.29	527	509.9	1.98	515	518.4	0.40	**514**	**515.9**	0.47	63.375
	6	424.9	436	440.3	1.30	435	444.7	2.01	432	434.8	0.52	**430**	**432.4**	0.59	63.375
	7	364.2	375	378.2	0.94	375	377.6	0.80	372	377.7	0.83	**370**	**372.1**	0.56	63.375
	8	318.7	335	338.4	1.18	334	335.9	0.44	327	334.2	0.90	**324**	**326.7**	0.72	63.375
P148	4	640.5	713	716.6	1.36	656	665.0	3.65	**641**	**642.0**	0.17	**641**	642.2	0.30	328.56
	5	512.4	567	568.5	0.58	523	532.5	4.42	**514**	**514.9**	0.18	**514**	515.1	0.33	328.56
	6	427.0	462	464.5	0.83	443	451.9	3.48	428	430.6	0.27	**428**	**429.6**	0.38	328.56
	7	366.0	395	396.6	0.43	384	387.0	1.47	368	369.2	0.21	**367**	**367.8**	0.26	328.56
	8	320.3	343	345.8	1.00	335	338.5	2.09	323	323.7	0.15	**322**	**322.6**	0.27	328.56
	9	284.7	305	306.9	0.64	299	302.2	1.80	287	289.6	0.53	**286**	**286.7**	0.35	328.56
	10	256.2	276	277.4	0.40	272	274.6	1.16	259	262.4	0.51	**258**	**260.1**	0.57	328.56
	11	232.9	248	248.8	0.29	245	249.4	1.80	237	238.5	0.42	**235**	**236.2**	0.37	328.56
	12	213.5	224	228.6	0.38	222	229.3	1.81	218	221.3	0.60	**216**	**217.4**	0.48	328.56
P205	4	2918.1	3513	3517.5	1.46	3016	3039.3	6.80	**2946**	**2965.6**	3.65	2953	2967.5	3.87	630.375
	5	2334.5	2799	2815.0	5.37	2474	2494.9	4.79	2364	2374.0	3.55	**2358**	**2364.1**	2.94	630.375
	6	1945.4	2353	2366.0	4.47	2060	2081.7	7.93	1984	2004.3	5.36	**1980**	**1983.5**	1.27	630.375
	7	1667.5	1982	1999.5	5.70	1795	1814.3	8.37	1709	1723.7	2.99	**1707**	**1718.4**	3.04	630.375
	8	1459.1	1750	1760.0	3.35	1560	1579.5	4.88	1507	1546.5	9.23	**1496**	**1511.6**	4.98	630.375
	9	1296.9	1545	1553.1	2.62	1368	1380.6	4.17	1337	1362.7	8.84	**1334**	**1346.2**	4.63	630.375
	10	1167.3	1385	1395.4	3.55	1247	1256.2	4.05	1189	1221.0	5.57	**1186**	**1210.3**	5.83	630.375
	11	1061.1	1254	1262.8	3.00	1131	1140.5	2.18	1095	1122.5	7.95	**1092**	**1114.3**	6.82	630.375
	12	972.7	1173	1176.0	1.02	1051	1066.3	6.12	1039	1061.1	3.45	**1012**	**1027.2**	4.49	630.375
	13	897.9	1064	1071.7	2.71	984	993.8	5.01	**944**	975.8	4.73	**944**	**955.4**	3.34	630.375
	14	833.8	998	1003.7	1.72	948	962.8	5.34	**944**	953.3	1.33	**944**	**944.0**	0.00	630.375

Note: Std = Standard deviation; LB = Lower bound

Figure 7 presents the Gantt chart for the best solution of P65 with four mated-stations obtained by the IABC algorithm. Figure 8 presents the Gantt chart for the best solution of P205 with 12 mated-stations obtained by the IABC algorithm.

As shown in Fig. 7, the value "4R" on the *Y*-axis refers to the right station of mated-station 4, which also indicates that the assembly line has eight stations. The number of rectangular boxes is the number of each task, and the order of the boxes indicates the precedence constraint of tasks in

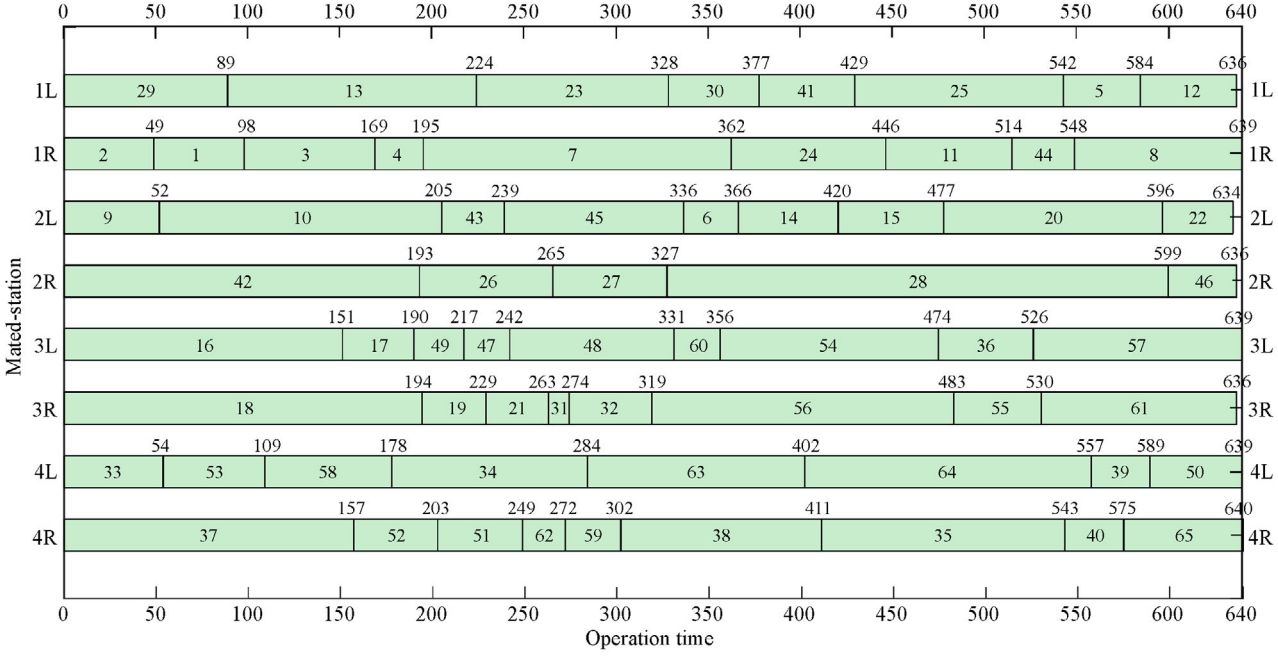

Fig. 7 Gantt chart for the best solution of P65 with four mated-stations

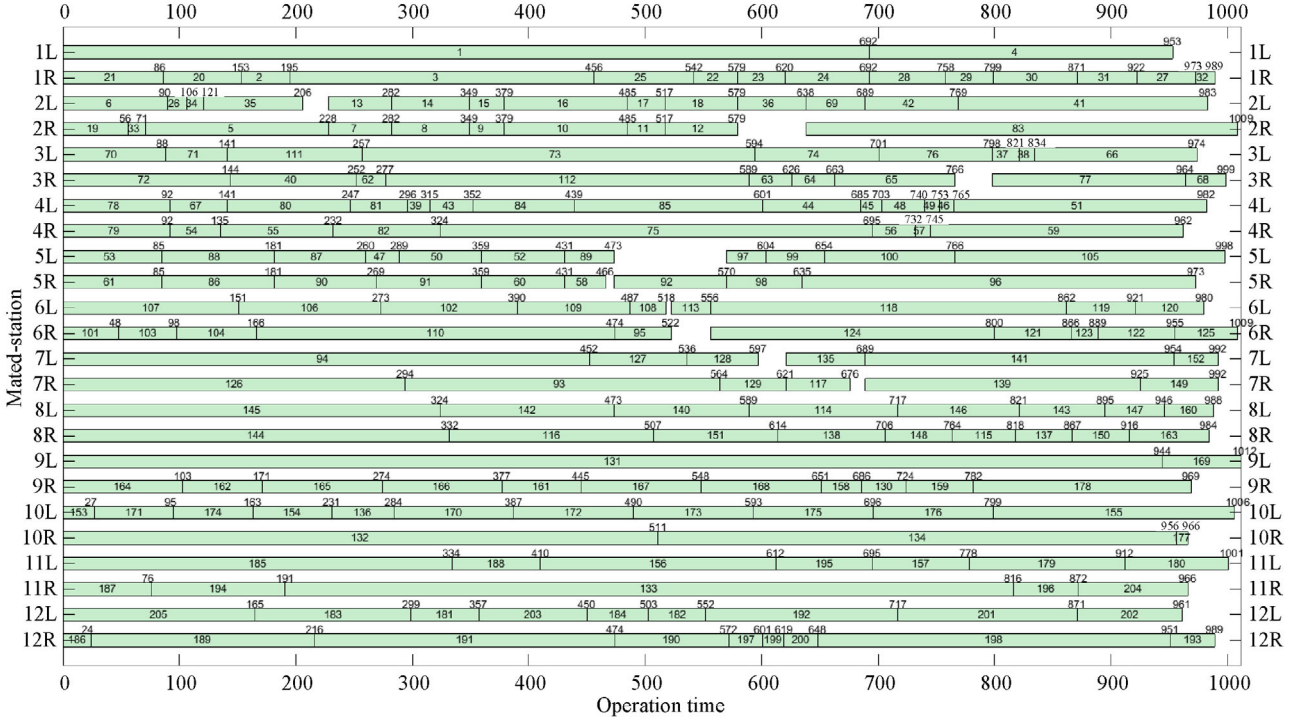

Fig. 8 Gantt chart for the best solution of P205 with 12 mated-stations

each station. For instance, these tasks of the station "4R" are assigned in the following order: 37–52–51–62–59–38–35–40–65.

The number at the upper-right corner of Task i refers to the finish time of the task. For each station, the finish time of the last task is the operation time. For example, the finish time of Task 65 is 640 units, which means that the operation time of the station "4R" is 640 units.

The cycle time is the maximum operation time of all the stations. Therefore, the cycle time of P65 with four mated-stations is 640 units.

The Friedman test is applied to analyze these results, in which the type of algorithm is considered a factor. Figure 9 presents the means of the average *RPD* of the best solutions among different algorithms. The best solutions obtained by IABC are significantly better than those by FFR, DABC, and *n*-GA. As shown in Fig. 10, the FFR and DABC algorithms obtain poor solutions on the average solutions, and the best solutions are obtained by the IABC algorithm. The *p*-value of the IABC algorithm is 0.01, which is smaller than 0.05. This result verifies that the proposed IABC is better than the other compared algorithms in terms of statistical significance.

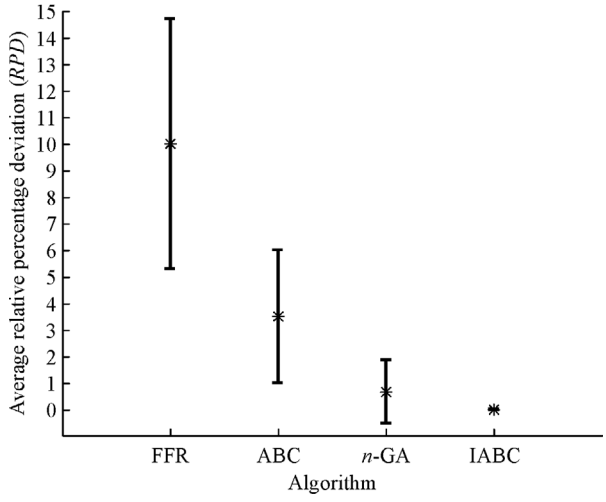

Fig. 9 Means of the average *RPD* of the best solutions and 95% least significant difference intervals for different algorithms

The convergence performance of these algorithms is further verified. The computational time is studied in Fig. 11 by solving P205 with 12 mated-stations. The IABC algorithm finds the best solution quickly, and it can converge fast. The comparisons of the convergence curve of the algorithms verify the search abilities of the employed bee strategy and onlooker bee strategy. Given the random strategy of scout bees, optimization is minimal with increasing time, which indicates that the scout bees can help the IABC algorithm escape from the local optima.

The foregoing analysis demonstrates that the proposed IABC algorithm is competitive with the other algorithms.

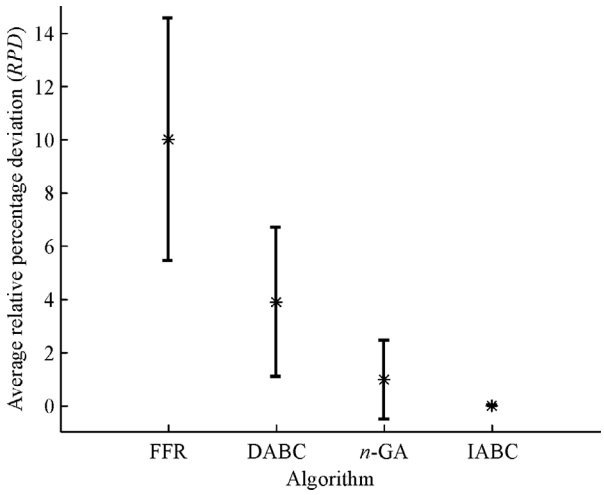

Fig. 10 Means of the average *RPD* of the average solutions and 95% least significant difference intervals for different algorithms

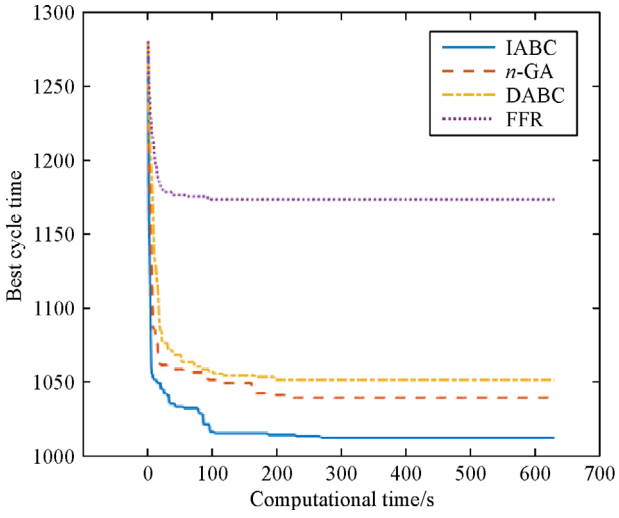

Fig. 11 Convergence curve for P205 with 12 mated-stations

6 Conclusions

In this paper, the TALBP is discussed. This research aims to minimize the cycle time of the two-sided assembly line for a given number of mated-stations. To solve the NP-hard problem efficiently, an improved ABC algorithm with the MaxTF heuristic rule is proposed in this paper. The MaxTF heuristic rule combined with the random rule can improve the quality and diversity of the solutions in population initialization. In the IABC algorithm, the employed and onlooker bees adopt the neighborhood strategy to enhance exploitation ability. By contrast, the scout bees generate a new solution that can prevent the algorithm from being trapped in local optima. The performance of the IABC algorithm is compared with that of other algorithms through various TALBP instances. Results show that the MaxTF rule can promote enhanced cycle time in the search

process. The results of the Friedman test of the best and average solutions for different problems show that the IABC algorithm is efficient and stable to solve the TALBP-2.

As a limitation, this model does not consider some realistic considerations, such as synchronous task constraints, probabilistic task times, different worker numbers of every workstation, and walking times of workers. Moreover, other objective optimizations can be considered, for example, maximizing the balancing efficiency, minimizing the total idle time, and maximizing the workload smoothness index. However, promoting the algorithm to solve realistic problems is critical in future research.

Acknowledgements This work was supported by the National Science and Technology Supporting Plan (Grant No. 2015BAF01B04). The authors would also like to thank the editor and the anonymous reviewers for their thorough reviews, detailed comments, and constructive suggestions, which helped improve the quality of this paper.

References

1. Scholl A, Becker C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 2006, 168(3): 666–693
2. Becker C, Scholl A. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research, 2006, 168(3): 694–715
3. Liu S B, Ng K M, Ong H L. Branch-and-bound algorithms for simple assembly line balancing problem. International Journal of Advanced Manufacturing Technology, 2008, 36(1–2): 169–177
4. Amen M. An exact method for cost-oriented assembly line balancing. International Journal of Production Economics, 2000, 64(1–3): 187–195
5. Sewell E C, Jacobson S H A. Branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS Journal on Computing, 2012, 24(3): 433–442
6. Mendes A R, Ramos A L, Simaria A S, et al. Combining heuristic procedures and simulation models for balancing a PC camera assembly line. Computers & Industrial Engineering, 2005, 49(3): 413–431
7. Kilincci O. A Petri net-based heuristic for simple assembly line balancing problem of type 2. International Journal of Advanced Manufacturing Technology, 2010, 46(1–4): 329–338
8. Zheng Q X, Li M, Li Y X, et al. Station ant colony optimization for the type 2 assembly line balancing problem. International Journal of Advanced Manufacturing Technology, 2013, 66(9–12): 1859–1870
9. Li Z, Kucukkoc I, Nilakantan J M. Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. Computers & Operations Research, 2017, 84: 146–161
10. Bartholdi J J. Balancing two-sided assembly lines: A case study. International Journal of Production Research, 1993, 31(10): 2447–2461
11. Kim Y K, Kim Y H, Kim Y J. Two-sided assembly line balancing: A genetic algorithm approach. Production Planning and Control, 2000, 11(1): 44–53
12. Wu E F, Jin Y, Bao J S, et al. A branch-and-bound algorithm for two-sided assembly line balancing. International Journal of Advanced Manufacturing Technology, 2008, 39(9–10): 1009–1015
13. Baykasoglu A, Dereli T. Two-sided assembly line balancing using an ant-colony-based heuristic. International Journal of Advanced Manufacturing Technology, 2008, 36(5–6): 582–588
14. Özcan U, Toklu B. Balancing of mixed-model two-sided assembly lines. Computers & Industrial Engineering, 2009, 57(1): 217–227
15. Kim Y K, Song W S, Kim J H. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers & Operations Research, 2009, 36(3): 853–865
16. Özcan U. Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. European Journal of Operational Research, 2010, 205(1): 81–97
17. Tapkan P, Ozbakir L, Baykasoglu A. Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. Applied Soft Computing, 2012, 12(11): 3343–3355
18. Khorasanian D, Hejazi S R, Moslehi G. Two-sided assembly line balancing considering the relationships between tasks. Computers & Industrial Engineering, 2013, 66(4): 1096–1105
19. Yuan B, Zhang C, Shao X. A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. Journal of Intelligent Manufacturing, 2015, 26(1): 159–168
20. Tang Q, Li Z, Zhang L. An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-II. Computers & Industrial Engineering, 2016, 97: 146–156
21. Li Z, Tang Q, Zhang L. Minimizing the cycle time in two-sided assembly lines with assignment restrictions: Improvements and a simple algorithm. Mathematical Problems in Engineering, 2016, 2016: 4536426
22. Tang Q, Li Z, Zhang L, et al. Balancing stochastic two-sided assembly line with multiple constraints using hybrid teaching-learning-based optimization algorithm. Computers & Operations Research, 2017, 82: 102–113
23. Li Z, Tang Q, Zhang L. Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study. Computers & Operations Research, 2017, 79: 78–93
24. Gansterer M, Hartl R F. One- and two-sided assembly line balancing problems with real-world constraints. International Journal of Production Research, 2017, (3): 1–18
25. Karaboga D. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report TR06. 2005
26. Mohammadi F G, Abadeh M S. Image steganalysis using a bee colony based feature selection algorithm. Engineering Applications of Artificial Intelligence, 2014, 31: 35–43
27. Liu Y, Liu S. A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. Applied Soft Computing, 2013, 13(3): 1459–1463
28. Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC)

algorithm. Journal of Global Optimization, 2007, 39(3): 459–471
29. Özcan U, Toklu B. A tabu search algorithm for two-sided assembly line balancing. International Journal of Advanced Manufacturing Technology, 2009, 43(7–8): 822–829
30. Moodie C L, Young H H. A heuristic method of assembly line balancing for assumptions of constant or variable work element times. Journal of Industrial Engineering, 1965, 16: 23–29
31. Tonge F M. Summary of a heuristic line balancing procedure. Management Science, INFORMs, 1960, 7(1): 21–42
32. Helgeson W B, Birnie D P. Assembly line balancing using the ranked positional weight technique. Journal of Industrial Engineering, 1961, 12: 394–398