Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration

Zeynel Abidin Çil [a], Zixiang Li [b,c,*], Suleyman Mete [d], Eren Özceylan [d]

[a] Department of Industrial Engineering, Izmir Democracy University, Izmir, Turkey
[b] Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan, Hubei, China
[c] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, Hubei, China
[d] Department of Industrial Engineering, Gaziantep University, Gaziantep, Turkey

A R T I C L E I N F O

Article history:
Received 5 November 2019
Received in revised form 17 March 2020
Accepted 9 May 2020
Available online 14 May 2020

Keywords:
Mixed-model assembly line
Human–robot collaboration
Bee algorithm
Artificial bee colony algorithm
Mixed-integer linear programming

A B S T R A C T

The collaboration of human workers and robots draws increasing attention from the manufacturing enterprises to embrace the Industry 4.0 paradigm in a competitive way. Motivated by the requirements of collaboration between human workers and robots in assembly lines, this study investigates the mixed-model assembly line balancing (MMALB) problem with the collaboration between human workers and robots. A mixed-integer linear programming (MILP) model is formulated to tackle the small-size problems optimally to minimize the sum of cycle times of models. Also, bee algorithm (BA) and artificial bee colony (ABC) algorithm are implemented and improved to solve the large-size problems due to the NP-hardness of this problem. The proposed BA algorithm utilizes a new employed bee phase to accelerate the evolution of the swarm and new scout phase to escape from being trapped into local optima and produce a high-quality and diverse population. The developed ABC proposes a new onlooker phase to accelerate the evolution of the whole swarm by removing the poor-quality solutions, new scout phase to achieve high-quality solutions while preserving the diversity of the swarm, and local search to enhance exploitation capacity. Computational study on a set of generated instances indicates that the improvements enhance the BA and ABC algorithm by a significant margin, and the proposed BA and ABC algorithm achieve competing performance in comparison with nine other algorithms, including the late acceptance hill-climbing algorithm, simulated annealing algorithm, genetic algorithm, particle swarm optimization algorithm, discrete cuckoo search algorithm, the original bee algorithm, and three artificial bee colony algorithms.

1. Introduction

Assembly lines are particular flow-based manufacturing systems, and they represent the last phase of production processes [1]. Assembly line balancing (ALB) problem, which assigns the tasks to a set of stations under specific constraints, such as cycle time constraint and precedence constraint, might occur during the design of manufacturing systems [2]. The essential changes are deriving from the differentiation of the market during the last decade, and constant pressure to increase the flexibility of the system due to increasing continuous customer-driven development of new products and a variety of customer demands. The traditional assembly line, which produces the single-model product, is only suitable to produce a type of standardized product [3]. Thus, different assembly system designs are critical to succeeding in productivity, quality, and customer satisfaction. It is a known fact that balancing the assembly lines considering mixed-model products solves the product variety problem, and it increases flexibility [4]. Besides alternative assembly line configurations, a significant opportunity to achieve the goals mentioned above is to improve the assembly systems utilizing new techniques like information, automation, virtualization of manufacturing processes, and communication technology, known as Industry 4.0. Industry 4.0 allows communication and interaction between human workers and machines. Moreover, it enables interactions between operators and machines to adapt to ever-increasing production variability, and this new type of interaction to have significant effects on the nature of the work [5].

* Corresponding author at: Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan, Hubei, China.
E-mail addresses: zabidin.cil@idu.edu.tr (Z.A. Çil), lizixiang@wust.edu.cn (Z. Li), smete@gantep.edu.tr (S. Mete), eozceylan@gantep.edu.tr (E. Özceylan).

https://doi.org/10.1016/j.asoc.2020.106394
1568-4946/© 2020 Elsevier B.V. All rights reserved.

Along with Industry 4.0, the collaboration of human workers and robots (machines), referred to as human–robot collaboration, has significant advantages in terms of quality, productivity, and customization in the assembly system [6]. In recent years, robots have shown their effectiveness in many areas of manufacturing. These robots could make the manufacturing process fast and reduce assembly costs. Robots are very efficient in performing repetitive tasks, and they can operate 24 h without fatigue, whereas the human workforce cannot show the same performance in such tasks. Therefore, the utilization of robots can help us increase productivity and provide consistent quality [7–9]. While robots cannot overcome product individualization in the manufacturing system, the human workforce can give much more flexibility for customization. It is an undeniable fact that robots cannot completely replace human workers. Nevertheless, combining of human workers and robots is the best way to eliminate possible drawbacks [10]. Although industrial robots are usually used in assembly lines, the close collaboration of these robots with human workers is not considered to be appropriate due to safety reasons. Collaborative robots (CoBots) are one of the Industry 4.0 operators to provide this cooperation in assembly lines [11].

According to the classification by Romero, et al. [11], CoBot is a type of industrial robot, which can perform a series of non-ergonomic and repetitive tasks. They have been specially considered to perform indirect close-collaboration with the smart human workers by means of safety (e.g. force sensing and collision) and intuitive interaction technologies, including easy shop-floor programming. They can provide advantages like increasing working space generally lost despite safety barriers and increasing the smart operator efficiency [5]. Overall, CoBots show benefits over traditional industrial robots in terms of safety, ease of programming, and ease of moving [10,12].

Considering the superiorities of CoBots mentioned above, allowing human workers and robots to collaborate in the same assembly line can be vital in the mixed-model line to ensure customization and simultaneous performance of human workers and robots (see Fig. 1). The MMALB problem with the interaction between human workers and robots should be thoroughly studied. However, there are limited related papers published. Specifically, Samouei and Ashayeri [13] take into account the MMALB problem for semi-automated operations, and they develop mathematical models to solve this problem. Weckenborg, et al. [14] propose a model and hybrid genetic algorithm for single-model ALB problem with collaborative robots. They formulate a mathematical model to solve the small-size instances optimally and develop a hybrid genetic algorithm to solve the large-size instances. Yaphiar, et al. [15] also consider MMALB problem for human–robot shared tasks and propose a mathematical model. In all the three papers, the type of robot is assumed only one so and robot assignment is not considered. However, in real applications, there are several different robots available, and it is necessary to consider the robot selection and assignment. And there is the only genetic algorithm applied by Weckenborg, et al. [14] and Samouei and Ashayeri [13] and Yaphiar, et al. [15] does not present the metaheuristic method for the large-size instances. Hence, it is also important to develop more methods to solve the large-size instance efficiently and effectively.

To overcome the above possible drawbacks and satisfy the demands of real industry, this study presents the first attempt to study the task assignment and selection and allocation of human workers and robots in the mixed-model assembly line. Notice that, from the previous literature analysis, and to the authors' best knowledge, this is the first time to study the problem with all the characters considered in this study. Specially, this study differentiates the published studies in the problem setting by taking into account of robot selection and assignment and other features and in the utilized method by developing two new metaheuristics for large-size problems. Also, this study mainly shows three contributions as follows. (1) A new mixed-integer linear programming (MILP) model is developed to minimize the total cycle time of models for a given station. Moreover, the proposed model is capable of solving the small-size problems optimally. (2) This study employs and improves the bee algorithm (BA) and artificial bee colony (ABC) algorithm to solve the large-size problems within the acceptable execution time due to the NP-hardness of this problem [9,16]. The proposed BA algorithm utilizes a new employed bee phase to accelerate the evolution of the swarm and new scout phase to escape from being trapped into local optima and produce a high-quality and diverse population. The developed ABC algorithm proposes a new onlooker phase to accelerate the evolution of the whole swarm by removing the poor-quality solutions, new scout phase to achieve high-quality solutions while preserving the diversity of the swarm, and local search to enhance exploitation capacity. (3) A set of nine other algorithms are also re-implemented to solve the considered problem, including the late acceptance hill-climbing algorithm, simulated annealing algorithm, genetic algorithm, particle swarm optimization algorithm, discrete cuckoo search algorithm, the original bee algorithm, and three artificial bee colony algorithms. Moreover, a comparative study is conducted to evaluate the performance of these algorithms. The computational study demonstrates that the improved algorithms outperform their original versions and achieve competing performance in comparison with other implemented algorithms.

The rest of the paper is organized as follows. Section 2 reviews the related studies, and Section 3 presents the problem definition and the proposed MILP model. Subsequently, Section 4 describes the proposed bee algorithms along with several improvements. Section 5 exhibits a general numerical example to highlight the features of the considered problem. Afterwards, Section 6 presents the computational experiments to evaluate the improvements and the proposed algorithms. Finally, Section 7 concludes this study and provides several directions for future research.

2. Literature review

As there is no paper studying the MMALB problem with physical human–robot collaboration, this section mainly reviews the most relevant and recent studies on MMALB problems in Section 2.1 and the studies on human–robot interaction in assembly operations in Section 2.2.

2.1. Review on mixed-model assembly line balancing problem

The pioneering works about MMALB problems are presented by Thomopoulos [16] and Thomopoulos [17]. After that, Gökċen and Erel [18] developed an integer programming model for the problem, and Erel and Gokcen [19] proposed a shortest-route formulation. Since then, many heuristic approaches are presented to solve the problem: variance algorithm [20], ant colony optimization algorithm [21,22], genetic algorithm [23], beam search algorithm [24], simulated annealing [25] and artificial bee colony algorithm [26], to cite just a few. The MMALB problem is also extended by considering more constraints and features. Specifically, the MMALB problem was combined with sequence-dependent setup time [27], buffer allocation [28,29], robotic system [24,30], U-type line [31], two-sided line [32,33] and parallel line [34]. In certain papers, the MMALB problem is considered with the sequencing problem [35,36]. The comprehensive review of the MMALB problem refers to the recent review paper by Battaïa and

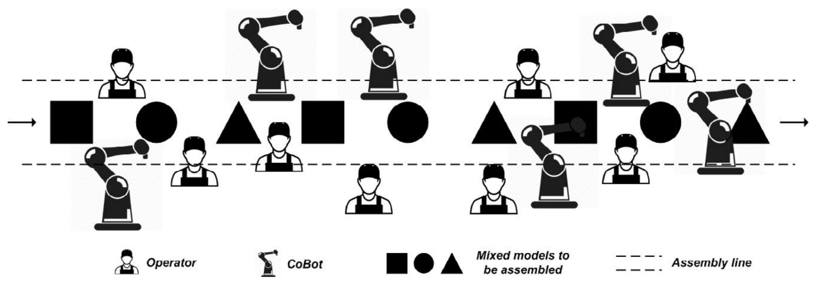


Fig. 1. Human–robots in manufacturing area.

Dolgui [37]. Among these published papers, most of the studies minimize the number of stations, and human–robot collaboration is still in need of further research. However, buffers, stochastic task time, different layouts, and others have already been discussed within the MMALB problem. In terms of solution approaches, simulated annealing algorithm, tabu search algorithm, genetic algorithm, and ant colony optimization algorithm have been mostly preferred by the researchers.

2.2. Review on human–robot interaction studies in assembly operations

Human–robot collaboration can be divided into four basic sections: (*i*) shared tasks and workspace with sequential, (*ii*) shared tasks and workspace jointly, (*iii*) the common task and workspace, and (*iv*) the common tasks and separate workspace [38,39]. Although there are many studies about human–robot interaction, there have been a small number of studies related to assembly or manufacturing cell.

In one of the first studies, Krüger, et al. [38] theoretically studied the cooperation of human–robot in assembly lines. This cooperation is divided into four different sections in terms of workplace and time-sharing systems. Lack of modelling is the main drawback of this study. Later, Tan, et al. [40] studied the collaboration of human workers and robots in cellular manufacturing by way of sharing tasks and workplaces. The primary objective of the study is to develop human–robot collaboration in cell production by the task modelling approach. Wallhoff, et al. [41] proposed a hybrid assembly station, where the robot can understand how the complete a task according to human instructions. In a later study, Chen, et al. [42] proposed a model for human and robot coordinated shared cell assembly. An optimal assembly strategy has been proposed to build a hybrid cell. A generalized stochastic petri net model is utilized to define all the potential states in the hybrid assembly cell. And Takata and Hirano [43] proposed the human–robot allocation method on-demand changing for hybrid assembly system to minimize the total production cost. More recently, Tsarouchi, et al. [10] studied human–robot collaboration for the implementation of collaborative tasks in hybrid assembly cells. Robot–human shares the same assembly cell, and one task is assigned to them concerning their capabilities. Also, an intelligent decision-making approach for human–robot task assignment is presented. The studies mentioned above indicate that human–robot interaction is an essential issue for a future factory where human capabilities and the robots' advantages are excellently integrated into a hybrid assembly cell.

Nevertheless, from the above literature review, there are limited papers on human–robot collaboration in the assembly line. Hence, this study presents the first attempt to study the MMALB problem with physical human–robot collaboration to minimize the sum of cycle times of models. This study mainly presents three contributions: (1) developing a new mathematical model to formulate this problem and solve the small-size problems optimally; (2) improving bee algorithms to solve the considered problem effectively; (3) implementing other nine algorithms and conducting a comprehensive evaluation study to evaluate the performance of the developed algorithms.

3. Problem definition and mathematical formulation

3.1. Problem definition

Assembly lines can be categorized into a robotic assembly line and manual assembly line in terms of operators (human workers and robots). The robotic assembly line is capable of assembling the products with higher rates, and this line is mainly utilized for small and medium production with low product variety [44]. On the other hand, the traditional manual assembly line is characterized by high labour costs, where at least a human worker is responsible for completing the tasks on one station.

With the development of robotic technologies, robots can increase the efficiency of the system and provide the highest level of safety for human workers simultaneously. And the human–robot collaboration draws increasing attention in the manufacturing system, especially in the assembly line to reduce capital cost and increase product variety and system flexibility. On the assembly line with human–robot collaboration, a robot and a human worker are both working in the same workplace in the assembly line to assemble the products. On this line, the simple and repetitive tasks are suitable for robots, and the complex tasks should be ideal for the human worker to improve system performance in terms of efficiency and flexibility [42,45].

The efficiency of the assembly line with human–robot collaboration depends on the task assignment and the selection and allocation of human workers and robots. Also, the ALB problem with collaborative robots comes up, which should be properly solved in other to achieve a high-quality assembly line with high efficiency and high flexibility. This study considers the MMALB problem with physical human–robot collaboration, where the main model assumptions are given as follows.

- ✓ All of the task operation times are known and deterministic.
- ✓ Operation times can be different according to the types of robots.
- ✓ Precedence diagrams are known and combined into a precedence diagram.
- ✓ The parallel working of a robot and a human worker is not considered. Namely, a robot and a human worker cannot perform different tasks on the same station simultaneously.
- ✓ Task division is not permitted.
- ✓ There are no limitations on the availability of all robots.
- ✓ The material transportation and set-up times are negligible.
- ✓ The number of stations is given and fixed.

3.2. Mathematical model

This section formulates the considered problem, where the utilized notations are first presented as follows.

i, j	Task ($i = 1, \ldots, I, j = 1, \ldots, I$)
k	Station ($k = 1, \ldots, K$)
r	Robots ($r = 1, \ldots, R$)
p, q	Operators ($p = 1, \ldots, P$). Here, P is equal to $R + 1$, $1, 2, \ldots, R$ corresponds to robots, and $R + 1$ corresponds to a human worker.
m	Model ($m = 1, \ldots, M$)
t_{mip}	Operation time of task i in model m for operator p
ts_{mi}	Starting time of task i in model m
SE_{ij}	1, if task i is assigned earlier than task j in the same station; 0, otherwise
$SEPE_{ij}$	1, if task i is assigned earlier than task j in the same station but they are performed by different operators; 0, otherwise
E	Set of pairs of tasks (i, j) in which task i is the immediate predecessor of task j
X_{ikp}	1, if task i is assigned to station k and performed by operator p; 0, otherwise
A_{kr}	1, if operator r is assigned to station k; 0, otherwise
C_m	Cycle time of each model
ψ	Large positive number
B_1	A large number which is calculated by using $B_1 > \max\limits_{m,p}(\sum\limits_i t_{mip})$ formula

Objective Function
The objective function (1) minimizes the sum of the cycle times of all models.

$$Min\ Obj = \sum_{m=1}^{M} C_m \tag{1}$$

Subject to
Constraint (2) indicates that each task must be assigned to one station and be performed by a human worker or a robot.

$$\sum_{k=1}^{K}\sum_{p=1}^{P} x_{ikp} = 1 \text{ for } i = 1, \ldots, I \tag{2}$$

Constraint (3) handles the precedence relations between tasks.

$$\sum_{k=1}^{K}\sum_{p=1}^{P} k * x_{ikp} - \sum_{k=1}^{K}\sum_{p=1}^{P} k * x_{jkp} \le 0 \text{ for } \ (i, j) \in E \tag{3}$$

Constraint (4) indicates that the sum of task operation times on the same station is less than cycle time.

$$\sum_{i=1}^{I}\sum_{p=1}^{P} t_{mip} * x_{ikp} \le C_m \quad \text{for } k = 1, \ldots, K, \text{and } m = 1, \ldots M \tag{4}$$

Constraint (5) ensures that the completion time of each task for all models must be less than or equal to the corresponding cycle time.

$$ts_{mi} + \sum_{p}^{P} t_{mip} \sum_{k=1}^{K} x_{ikp} \le C_m \quad \text{for } m = 1, \ldots M \text{ and } i = 1, \ldots, I \tag{5}$$

As more than one operator can be assigned to the station, scheduling within the station is required. For a pair of tasks with precedence relations, constraint (6) handles the situation when they are allocated to the same station and are performed by the same operator and constraint (7) handles the situation when they are allocated to the same station and are performed by different operators.

$$ts_{mi} + t_{mip} * x_{ikp} + \psi.\left(x_{ikp} + x_{jkp} - 2\right) \le ts_{mj} \\ \text{for } k = 1, \ldots, K,\ p = 1, \ldots, R+1, \\ m = 1, \ldots, M, i < j \text{ and } (i, j) \in E \tag{6}$$

$$ts_{mi} + t_{mip} * x_{ikp} + \psi.\left(x_{ikp} + \textstyle\sum_{q \in Pset, q \ne p} x_{jkq} - 2\right) \le ts_{mj} \\ \text{for } k = 1, \ldots, K,\ m = 1, \ldots, M,\ p = 1, \ldots, R+1, i < j \text{ and} \\ (i, j) \in E \tag{7}$$

For a pair of tasks which do not have precedence relation, constraint (8) and constraint (9) deal with the situation when they are allocated to the same station and are performed by the same operator. And constraint (8) takes effect when task i starts before task j; constraint (9) takes effect, otherwise.

$$ts_{mi} + t_{mip} * x_{ikp} - 2 * \psi.\left(2 - x_{ikp} - x_{jkp}\right) \\ \le ts_{mj} + B_1.\left(1 - SE_{ij}\right) \text{ for} \\ k = 1, \ldots, K, m = 1, \ldots, M, p = 1, \ldots, R+1 \text{ and} \\ i, j = 1, \ldots, I, J\colon i < j \wedge (i, j) \notin E \tag{8}$$

$$ts_{mj} + t_{mjp} * x_{jkp} - 2 * \psi.\left(2 - x_{ikp} - x_{jkp}\right) \\ \le ts_{mi} + B_1.SE_{ij} \text{ for} \\ k = 1, \ldots, K, m = 1, \ldots, M, p = 1, \ldots, R+1 \text{ and} \\ i, j = 1, \ldots, I, J\colon i < j \wedge (i, j) \notin E \tag{9}$$

For a pair of tasks which do not have precedence relation, constraint (10) and constraint (11) deal with the situation when they are allocated to the same station and are performed by different operators. Similar to the previous situation, constraint (10) takes effect when task i starts before task j; constraint (11) takes effect, otherwise.

$$ts_{mi} + t_{mip} * x_{ikp} + \psi.\left(x_{ikp} + \sum_{q \in P_{set}, q \ne p} x_{jkq} - 2\right) \\ \le ts_{mj} + B_1.\left(1 - SEPE_{ij}\right) \text{ for} \\ k = 1, \ldots, K, m = 1, \ldots, M, p = 1, \ldots, R+1 \text{ and} \\ i, j = 1, \ldots, I, J\colon i < j \wedge (i, j) \notin E \tag{10}$$

$$ts_{mj} + \sum_{q \in P_{set}, q \ne p} t_{mjq} * x_{jkq} + \psi.\left(x_{ikp} + \sum_{q \in P_{set}, q \ne p} x_{jkq} - 2\right) \\ \le ts_{mi} + B_1.SEPE_{ij} \text{ for} \\ k = 1, \ldots, K, m = 1, \ldots, M, p = 1, \ldots, R+1 \text{ and} \\ i, j = 1, \ldots, I, J\colon i < j \wedge (i, j) \notin E \tag{11}$$

Constraint (12) indicates that the starting time of each task must be equal to or greater than zero.

$$ts_{mi} \ge 0 \text{ for } i = 1, \ldots, I \text{ and } m = 1, \ldots, M \tag{12}$$

Constraint (13) ensures that, if a task is assigned to station k and performed by operator p, operator p must be assigned to station k.

$$\sum_{i=1}^{I} x_{ikp} - \psi * A_{kp} \le 0 \text{ for } k = 1, \ldots, K \textit{ and } p = 1, \ldots, R+1 \tag{13}$$

Constraint (14) indicates that only a robot can be assigned to a station.

$$\sum_{r} A_{kr} \le 1 \text{ for } k = 1, \ldots, K \tag{14}$$

Constraint (15) indicates binary and integer variables.

$$A_{kr}, x_{ikp} \in \{0, 1\}, C_m \ge 0 \text{ and integer} \tag{15}$$

4. Proposed methodologies

Metaheuristics have been in many global optimization problems [46–51] due to easy implementation and competing performance. Due to the complexity of the considered problem as known NP-hard [9,16], this study suggests and improves two population-based algorithms, bee algorithm (BA) and artificial bee colony algorithm (ABC), to tackle the large-size problems. These two methods are selected mainly due to their unique optimization mechanisms, which assist the two algorithms in achieving competing performances in solving many other discrete optimization problems [52–54]. Specifically, both algorithms utilize the scouts to explore new solutions and hence increase the exploration capacity and help the two algorithms to escape from being trapped into local optima. Another reason for selecting the two methods is that they suit the considered problem. As you will see in Section 4.1, the task assignment vector and robot allocation vector are utilized to obtain a feasible solution, and it is more efficient to optimize these two factors with local search rather than crossover operators or others in the preliminary experiments. And the selected methods are both population algorithms based on local search, and hence they suit the considered problem.

To tackle the considered problem effectively, the proposed BA utilizes a new employed bee phase to accelerate the evolution of the swarm and new scout phase to escape from being trapped into local optima and produce a high-quality and diverse population. The developed ABC proposes a new onlooker phase to accelerate the evolution of the whole swarm by removing the poor-quality solutions, new scout phase to achieve high-quality solutions while preserving the diversity of the swarm, and local search to enhance exploitation capacity. All these improvements are in favour of BA and ABC algorithm achieving a proper balance between exploitation and exploration. The following sections first introduce the proposed encoding and decoding and afterwards provide the details of the proposed two algorithms along with some improvements.

4.1. Solution presentation

The considered problem involves two interacted sub-problems: (1) the assignment of tasks and (2) the selection and allocation of human workers and robots. The mixed-model situation makes this problem even more complicated, where the best robot/worker for one station cannot be determined simply. Hence, this study utilizes two encoding vectors: task assignment vector and robot allocation vector. The task assignment vector corresponds to the allocated stations of tasks, and the robot allocation vector corresponds to the allocated robots to stations. Figs. 2 and 3 show the illustrated encoding scheme, for instance, with 11 tasks, two models, and four available robot types, and also the same instance is analysed in Section 5 for the understanding of solution procedures. For example, task 1, task 2, and task 3 are allocated to station 1, station 2, and station 1 (see Fig. 2), respectively. On the other hand, robot 4 and robot 3 are allocated to station 1 and station 2, respectively (see Fig. 3). Whether a human worker is not determined by these two vectors, but it is determined during the decoding procedure.

The decoding procedure is presented as follows, where *nt* is the number of tasks, *ns* is the number of stations, *nm* is the number of models. In Step 1, for any pair of tasks with precedence relation, if the predecessor is allocated to the latter station, the corresponding stations of these two tasks are exchanged. Besides, this procedure terminates when all the predecessors of any task are allocated to the former or the same station, ensuring that the precedence constraint is satisfied. In Step 2, three types of assembly alternatives are tested: only one robot, only one human worker and one robot, and one human worker. Here, when both one robot and one human worker are allocated to one station, one task on this station is performed by the robot or human worker which can complete this task with less operation time.

4.2. Original bee algorithm

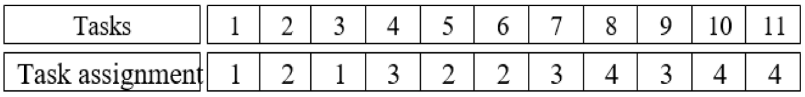
The main procedure of the original BA, referred to as OBA, is illustrated in Algorithm 1 [55]. And the OBA has five parameters: the number of scouts (S), the number of employed bees (P), the number of best-employed bees (e), the number of bees for the best e individuals (nep), the number of bees for the P-e individuals (nsp). OBA starts with initializing the S scout bees randomly, and subsequently, the main loop comprised of updating the best-employed bees, updating the remained employed bees, and updating the scouts is repeated until a termination criterion is satisfied.

Within the loop, the scout bees are sorted in the increasing order of the fitness values at first. Afterwards, for each individual in the best e individuals, a set of nep neighbour solutions are achieved, and the incumbent individual is replaced with the best neighbour solution when the better fitness is obtained. Subsequently, for the individual in the remained P-e employed bees, a set of nsp neighbour solutions are obtained, and the incumbent individual is replaced with the best neighbour solution when the better fitness is obtained. Finally, for the remained scouts, they are replaced with randomly generated solutions.

4.3. The proposed improved bee algorithm

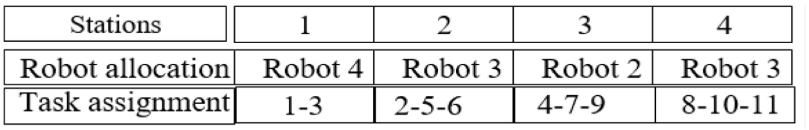
Nevertheless, there are two possible weaknesses when utilizing OBA to solve the considered problem. (1) Many randomly generated solutions have poor quality for large-size problems. (2) The best e individuals are always preserved even when no improvement is achieved, and the OBA might be trapped into local optima. Hence, this study presents an improved BA, referred to as IBA, in this section, and the procedure of the IBA is presented in Algorithm 2, where one more parameter *limit* is utilized. The procedure of the IBA is similar to that of OBA, but the main segments are quite different.

The main differences between the IBA and OBA are clarified as follows. (1) In Step 2, the fitness values of duplicated individuals are set to a large positive number. Namely, the duplicated individuals will not be preserved. This modification helps avoid the algorithm' being trapped into local optima. (2) In Step 3 and Step 4, new acceptance criterion is applied, and the incumbent individual is replaced with the new individual immediately when the same or better fitness value is obtained. The modification accelerates the evolution of the swarm and avoids the wasted time to search around poor solutions. (3) In Step 5, both the worst individuals and the individuals from the best e individuals, which have not been improved in *limit* consecutive iterations, are abandoned. Here, the abandonment of the individuals which cannot be further enhanced helps the algorithm to escape from being trapped into local optima. To have a high-quality new solution, this study replaces the incumbent solution with the best one from a set of neighbour solutions (set to 10) achieved by conducting neighbour operators for several times (set to (1). The proposed method to obtain a new solution aims at producing a high-quality and diverse population. In short, all these improvements are in favour of IBA achieving a proper balance between exploitation and exploration.



Tasks	1	2	3	4	5	6	7	8	9	10	11
Task assignment	1	2	1	3	2	2	3	4	3	4	4

Fig. 2. Illustrated encoding scheme for task assignment.



Stations	1	2	3	4
Robot allocation	Robot 4	Robot 3	Robot 2	Robot 3
Task assignment	1-3	2-5-6	4-7-9	8-10-11

Fig. 3. Illustrated encoding scheme for robot allocation.

```
Input: Instance data, task assignment vector and robot allocation vector
          % Adjust the task assignment to satisfy the precedence constraint
          Do
            Terminate=1;
            For i:=1 to nt-1 do
              For j:=i+1 to nt do
              If (task i is the predecessor of task j and task i has the larger values in the task
Step 1:       assignment vector)
              Exchange the values of task i and task j in the task assignment vector;
              Set Terminate=0;
              Endif
            Endfor
            Endfor
          Until (Terminate=1)
          For k:=1 to ns do
            Achieve the robot r allocated to station k based on robot allocation vector;
            Achieve the tasks whose values in the task assignment vector are equal to k;
            Calculate the completion time when only utilizing robot r;
Step 2:     Calculate the completion time when only utilizing one human worker;
            Calculate the completion time when utilizing one robot and human worker;
            Select the minimum completion time and the corresponding robot/worker as the
            selected robot/worker for this station.
          Enfor
          For m:=1 to nm do
            Calculate the cycle time $C_m$ of model m which is the maximum value of the completion
Step 3:     times of tasks for model m;
          Endfor
          Calculate the sum of the cycle times of models;
Output: the total cycle time of models
```

4.4. Original artificial bee colony algorithm

The main procedure of the original ABC, referred to as OABC, is illustrated in Algorithm 3, where *Fit* (p) is the fitness value of individual p. OABC has two parameters: the number of employed bees or onlooker bees (S) and the number of $(limit)$ trials after which a food source is abandoned [56–58]. OABC starts with initializing several S individuals randomly, and afterwards, the employed bee phase, the onlooker phase, and the scout phase are conducted and repeated until a termination criterion is met.

In employed bee phase, a neighbour solution of each individual is generated by one employed bee, and greedy acceptance is utilized to determine whether to replace the incumbent one. In the onlooker phase, one onlooker selects one individual using roulette wheel selection, where the probability of a food source to be selected is calculated with $\mathrm{Prob}\,(p) = \frac{1/Fit\,(p)}{\sum_{k=1}^{k=PS} 1/Fit\,(k)}$. Subsequently, the onlooker obtains a new neighbourhood food source, and this new food source replaces the incumbent one when better fitness is achieved. In the scout phase, one nectar of food source, which cannot be improved for a predetermined number of times, is selected. Then, this selected food source is abandoned, and the scout generates a new food source randomly to replace the abandoned one.

4.5. The proposed improved artificial bee colony algorithm

To enhance the exploitation capacity and achieve the high-quality solution to replace the abandoned solutions, this section presents an improved ABC algorithm, referred to as IABC algorithm, where r is a new parameter within [0, 1]. The main procedure of the proposed IABC is illustrated in Algorithm 4. The proposed IABC follows the main procedure of the OABC, but the main segments of the IABC are quite different from that in OABC.

To differentiate the proposed IABC from OABC, the main differences are clarified as follows. (1) In the onlooker phase, one individual is selected utilizing the binary tournament selection. And the best and non-duplicated S individuals, from the incumbent solutions and the neighbour solutions ($2 \times S$ individuals in total), are selected as the new swarm. This modification accelerates the evolution of the whole swarm by removing the poor-quality solutions, and hence the IABC mainly searches around the most promising search space. (2) In the scout phase, the abandoned solution is replaced with the best one from a set of neighbour solutions (set to 10) achieved by conducting neighbour operators for several times (set to (1). This modification aims at achieving high-quality solutions while preserving the diversity of the swarm. (3) Optionally, IABC utilizes a local search to enhance

Algorithm 1: Main procedure of the OBA algorithm

```
Step 1:  % Initialization
         Initialize the $S$ scout bees randomly;
         Do
         % Sort scout bees
Step 2:  Sort the scout bees in the increasing order of the fitness values, and select
         best $P$ individuals as employed bees and select best $e$ employed bees;
         % Update the best employed bees
         For $p$:=1 to $e$ do
           Obtain a set of $nep$ neighbor solutions of individual $p$;
Step 3:    Select the best neighbor solution $p'$ ;
           $p \leftarrow p'$ when $\mathrm{Fit}(p') < Fit(p)$;
         Endfor
         % Update the remained employed bees
         For $p$:=$e$+1 to $P$ do
           Obtain a set of $nsp$ neighbor solutions of individual $p$;
Step 4:    Select the best neighbor solution $p'$ ;
           $p \leftarrow p'$ when $\mathrm{Fit}(p') < Fit(p)$;
         Endfor
         % Update scouts
         For $p$:=$P+1$ to $S$ do
Step 5:    Obtain a solution $s'$ randomly and update $p$ with $p \leftarrow s'$;
         Endfor
         Until (termination criterion is satisfied)
Output: Objective value
```

Algorithm 2: Main procedure of the proposed IBA algorithm

```
Step 1:  % Initialization
         Initialize the $S$ scout bees randomly;
         Do
         % Sort scout bees
Step 2:  Set the fitness values of duplicated individuals as a large positive number;
         Sort the scout bees in the increasing order of the fitness values, and select best $P$
         individuals as employed bees and select best $e$ employed bees;
         % Update the best employed bees
         For $p$:=1 to $e$ do
           For $k$:=1 to $nep$ do
Step 3:      Obtain one neighbor solution $p'$ of individual $p$;
             $p \leftarrow p'$ when $\mathrm{Fit}(p') \leq \mathrm{Fit}(p)$;
           Endfor
         Endfor
         % Update the remained employed bees
         For $p$:=$e$+1 to $P$ do
           For $k$:=1 to $nsp$ do
Step 4:      Obtain one neighbor solution $p'$ of individual $p$;
             $p \leftarrow p'$ when $\mathrm{Fit}(p') \leq \mathrm{Fit}(p)$;
           Endfor
         Endfor
         % Update scouts
         For $p$:=$P+1$ to $S$ do
           Select one individual $s$ from the best $P$ individuals;
           Obtain a set of neighbor solutions of this selected individual by conducting neighbor
           operators for several times;
           Select the best neighbor solution $s'$, which is different from the $s$, and set $p \leftarrow s'$;
         Endfor
Step 5:  For $p$:=1 to $e$ do
           If (individual $p$ has not been improved in $limit$ consecutive iterations)
             Obtain a set of neighbor solutions of individual $p$ by conducting neighbor operators
             for several times;
             Select the best neighbor solution $p'$, which is different from the $p$, and set $p \leftarrow p'$;
           Endif
         Endfor
         Until (termination criterion is satisfied)
Output: Objective value
```

```
Algorithm 3: Main procedure of the OABC algorithm
Step 1:   Initialize the $S$ individuals randomly;
          Do
          % Employed bee phase
          For $p$:=1 to $S$ do
Step 2:     Obtain one neighbor solution $p'$ of individual $p$;
            $p \leftarrow p'$ when $\mathrm{Fit}(p') < Fit(p)$;
          Endfor
          % Onlooker phase
          For $p$:=1 to $S$ do
            Obtain the probability value for each individual;
Step 3:     Select one individual $s$ using roulette wheel selection;
            Obtain one neighbor solution $s'$ of individual $s$;
            $s \leftarrow s'$ when $\mathrm{Fit}(s') \leq \mathrm{Fit}(s)$
          Endfor
          % Scout phase
Step 4:   Select one solution that has not been improved for consecutive limit times;
          If exists, replaced the selected solution with a randomly generated solution;
          Until (termination criterion is satisfied)
Output: Objective value
```

exploitation where rand [0, 1] is a random number within 0 and 1, r is a new parameter needed to be determined, and Nt is the number of tasks. Namely, the local search is conducted for one individual with a pre-determined probability. In summary, the modifications enhance the exploitation capacity of the proposed IABC and help IABC to achieve a proper balance between exploitation and exploration.

4.6. Utilized neighbour operators

The proposed IBA and IABC utilize two neighbour operators, swap operator, and mutation operator, to obtain high-quality neighbour solutions. For the task assignment vector, the swap operator selects two tasks with different stations randomly and then exchanges the corresponding stations of these two tasks. The mutation operator selects a task randomly and replaces the corresponding station of this task with a different station. For the robot selection vector, the swap operator selects two stations randomly and exchanges the robots on these two stations. The mutation operator selects a station randomly and replaces the corresponding robot with a different robot. For an individual, the task assignment vector or the robot allocation vector is first selected randomly. Afterwards, the swap operator or the mutation operator is chosen randomly to modify the selected vector. Here, the utilization of two neighbour operators aims at expanding the search space. An example of utilizing the two neighbour operators to alter the task assignment vector is depicted in Fig. 4. In the figure, swap operator selected randomly task1 and task 2, which are assigned station 1 and station 2, then exchange the stations of these two selected tasks. Similarly, the mutation operator selects task 4 as randomly, which is assigned station 3, then replaces station 3 of the task with station 2.

5. An illustrated example

This section shows an example with 11 tasks to highlight the main features of the considered problem. The combined precedence diagram of this instance is presented in Fig. 5. Here, one task can be allocated only when all its predecessors have been allocated, e.g. task 7 cannot be performed until task 3, task 4, and task 5 have been completed. The operation times by robots and human workers are illustrated in Table 1. In this table, NA donates that this task cannot be performed by the corresponding robot or worker, e.g. task 1 in model 1 cannot be performed by a human worker.

Table 1
Task operation times by robots and human worker.

Model	Task	Robot 1	Robot 2	Robot 3	Robot 4	Worker
1	1	103	55	58	59	NA
	2	NA	NA	NA	NA	79
	3	94	105	110	85	NA
	4	NA	NA	NA	NA	85
	5	62	81	37	141	NA
	6	66	88	24	42	NA
	7	NA	NA	NA	NA	20
	8	NA	NA	NA	NA	0
	9	90	36	91	26	NA
	10	43	73	20	20	NA
	11	NA	NA	NA	NA	100
2	1	71	119	26	61	NA
	2	NA	NA	NA	NA	85
	3	63	89	76	90	NA
	4	NA	NA	NA	NA	0
	5	62	90	37	135	NA
	6	65	24	50	31	NA
	7	NA	NA	NA	NA	169
	8	NA	NA	NA	NA	65
	9	74	26	31	127	NA
	10	75	134	26	31	NA
	11	NA	NA	NA	NA	60

Table 2 illustrates the detailed decoding procedure utilizing the encoding scheme in Figs. 2 and 3. Here, the first two rows present the task assignment in Fig. 2 and robot allocation based on the encoding scheme presented in Fig. 3. Notice that the task assignment in the encoding scheme provides the detailed task assignment directly, whereas the selection of robot or worker still needs to be determined.

An example of selecting the robot and worker on station 2 is presented in Table 3, where NA means that there is at least one task cannot be performed by this assembly alternative. As either robot 3 or a worker cannot complete the tasks on station 2, robot 3 and a worker are allocated to this station. After the selection of robot and worker, the detailed task assignment and allocation of robots and human worker in Table 2 are: robot 4 is allocated to station 1 to perform task 1 and task 3, robot 3 and human worker is allocated to station 2 to perform task 2, task 5 and task 6, robot 2 and human worker is allocated to station 3 to perform task 4, task 7 and task 9 and finally robot 3 and human worker is allocated to station 4 to perform task 8, task 10 and task 11. And the seventh and eighth rows clarify whether one task is performed by a robot or a worker, e.g. task 5 and

Algorithm 4: Main procedure of the proposed IABC algorithm

```
Step 1:  Initialize the $S$ individuals randomly;
         Do
         % Employed bee phase
         For $p$:=1 to $S$ do
Step 2:    Obtain one neighbor solution $p'$ of individual $p$;
           $p \leftarrow p'$ when $\mathrm{Fit}(p') \le \mathrm{Fit}(p)$;
         Endfor
         % Onlooker phase
         For $p$:=1 to $S$ do
           Select an individual $s$ with binary tournament selection;
Step 3:    Obtain one neighbor solution $s'$ of individual $s$;
         Endfor
         Select the best and non-duplicated $S$ individuals from the incumbent solutions and the
         neighbor solutions ($2 \times S$ individuals in total);
         % Scout phase
         For $p$:=1 to $S$ do
           If (individual $p$ has not been improved in limit consecutive iterations)
             Obtain a set of neighbor solutions of individual $p$ by conducting neighbor operators
Step 4:      for several times;
             Select the best neighbor solution $p'$, which is different from the $p$, and set $p \leftarrow p'$;
           Endif
         Endfor
         % Local search
         For $p$:=1 to $S$ do
           If rand[0,1] $\le r$
             For $k$:=1 to Nt do
Step 5:        Obtain one neighbor solution $p'$ of individual $p$;
               $p \leftarrow p'$ when $\mathrm{Fit}(p') \le \mathrm{Fit}(p)$;
             Endfor
           Endif
         Endfor
         Until (termination criterion is satisfied)
Output: Objective value
```

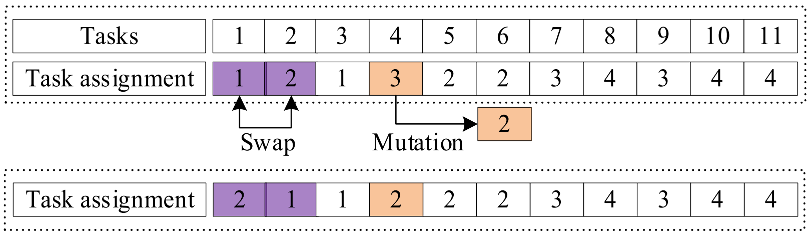


Fig. 4. The proposed neighbour operators.

task 6 is performed by robot 3 and task 2 is performed by the human worker on station 2. The remained rows in Table 2 show the calculation of the total cycle time, which is the sum of the cycle times of the two models.

For better vision, the schematic representation of the illustrative example is also provided in Fig. 6. As you can see, the main feature of the considered problem is that a robot and a human worker can work on one same station and perform the assigned tasks in sequence, which differentiates the considered problem from that published.

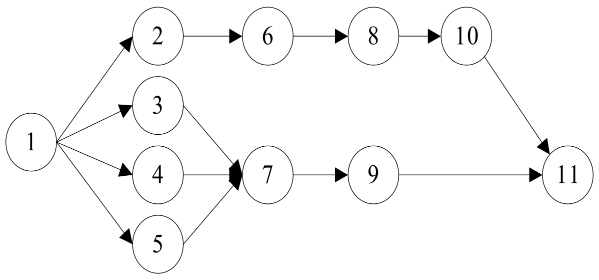


Fig. 5. Combined precedence diagram of the illustrated example.

6. Computational analysis

This section tests the performance of the formulated model and developed algorithms. Since there is no published paper studying the considered problem, this section first generates a set of instances in Section 6.1. Subsequently, the proposed improvements are evaluated in Section 6.2, where the proposed two algorithms are compared with their original versions. Finally, Section 6.3 presents the comparative study where the proposed mathematical model and bee algorithms are analysed and compared with several other algorithms.

6.1. Experiment setting

As there is no published paper studying the considered problem, this section first generates a set of 11 instance sets, ranging from the small-size instance set with seven tasks to the large-size instance set with 148 tasks. For these instance sets, eleven precedence relations (available at http://www.assembly-

Table 2
Detailed task assignment and robot/worker allocation.

		Station 1	Station 2	Station 3	Station 4
Encoding scheme	Task assignment	1, 3	2, 5, 6	4, 7, 9	8, 10, 11
	Robot allocation	Robot 4	Robot 3	Robot 2	Robot 3
Decoding procedure	Task assignment after adjusting the encoding	1, 3	2, 5, 6	4, 7, 9	8, 10, 11
	Robot allocation in the encoding	Robot 4	Robot 3	Robot 2	Robot 3
	Worker determined utilized the method of selection of robot and human worker in Table 3	No	Yes	Yes	Yes
	Tasks performed by robot	1, 3	5, 6	9	10
	Tasks performed by worker	–	2	4,7	8, 11
	Task operation times in Model 1	59, 85	79, 37, 24	85, 20, 36	0, 20, 100
	Task operation times in Model 2	61, 90	85, 37, 50	0, 169, 26	65, 26, 60
Calculate the objective function	Completion time of stations in Model 1	144	140	141	120
	Completion time of stations in Model 2	151	172	195	151
	Cycle time of Model 1	144			
	Cycle time of Model 2	195			
	Total cycle time	339			

Table 3
The selection of robot and human worker.

Assembly types	Assigned tasks	Completion time of models	Selected assembly type
Robot 3	2, 5, 6	NA/NA	Robot 3 and a worker
Worker	2, 5, 6	NA/NA	
Robot 3 and a worker	2, 5, 6	140/172	

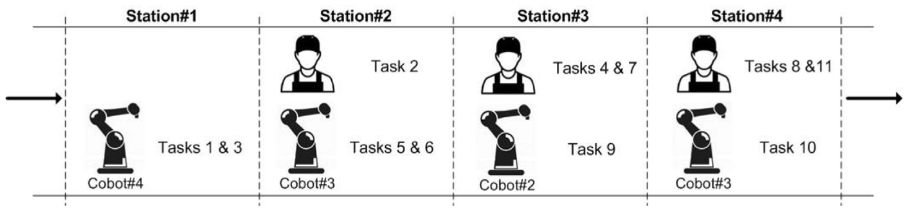


Fig. 6. Schematic representation of the illustrative example.

line-balancing.de/) are selected as the combined precedence diagram for the MMALB problem. Also, different predetermined numbers of stations are considered for these 11 problem sets, leading to a total number of 41 instances. During the generation of the test instances, two models and four robot types are considered for each instance. And the task times of each model are generated according to uniform distribution for human workers and robots, where task times of human worker and robots range within 15 and 200. Notice that different robots can perform a task with different operation times as they have different properties. Detailed precedence relations and task operation times of the tested instances are available upon request.

As the proposed MILP model cannot achieve satisfying solutions when solving large-size instances, this model mainly tackles the small-size instances with less than 53 tasks. And this model is solved utilizing the CPLEX solver in the General Algebraic Modelling System 23.0, and the CPLEX solver terminates when the optimal solution is found and verified or the computation time reaches 3600 s (s). To evaluate the performance of the proposed algorithms, a set of known algorithms are re-implemented in this paper late acceptance hill-climbing algorithm (LAHC), simulated annealing algorithm (SA), genetic algorithm (GA), particle swarm optimization (PSO) algorithm, discrete cuckoo search algorithm (DCS), the original bee algorithm (OBA), the original artificial bee colony algorithm (OABC) and two artificial bee colony algorithms (ABC1 and ABC2) [59]. Here, ABC1 and ABC2 achieve competing for performance in comparison with 11 algorithms when solving the robotic two-sided assembly line balancing problems [59]. Detailed procedures of these tested algorithms and the utilized parameter values calibrated with the full factorial experimental design as that in Tang, et al. [53] are available upon request.

All the tested algorithms terminate when the elapsed running time reaches Nt $\times$ Nt $\times$ τ milliseconds for one instance, where Nt is the number of tasks for this tested instance. To have a better observation of algorithms' performance under different computation time, the value of τ is set to 20, 40, 60, 80 and 100; respectively. And all the instances described above are solved by the implemented algorithms for ten repeated times to have more data for statistical analysis. As different instances are solved, the relative percentage deviation (RPD) is applied to transfer the obtained cycle times utilizing $RPD = 100 \cdot (CT_{some} - CT_{Best}) / CT_{Best}$, where CT_{some} is the achieved cycle time by one algorithm in one run and CT_{Best} is the minimum cycle time by all the algorithms in ten runs.

6.2. Evaluating the improvements

This section first tests the performances of the improvements on IBA and IABC, and two comparative studies are conducted as follows. For IBA, it is compared with three original BA algorithms: BA-O1 where all the parameters are set as their original values in Section 4.2, BA-O2 where the original employed bee phase is conducted and BA-O3 where the original scout phase is conducted. For IABC, it is compared with three original ABC algorithms: ABC-O1, where all the parameters are set as their original values in Section 4.4, ABC-O2 where the original scout phase is conducted and ABC-O3 where no local search is conducted. Notice that the remained parameters are set to the same values for these tested algorithms.

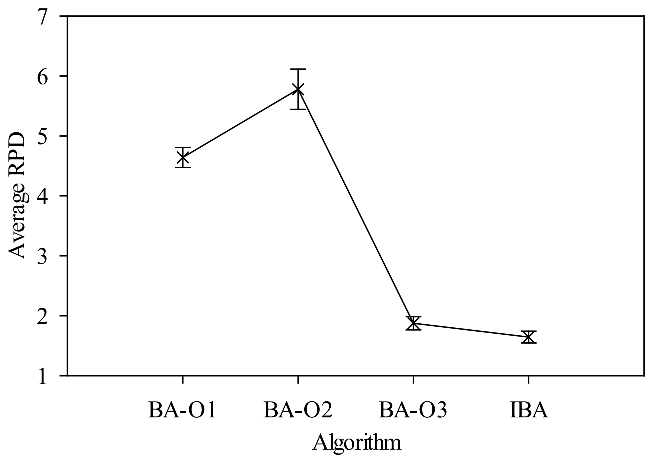


Fig. 7. Means plot and 95% Tukey HSD confidence intervals for BA algorithms.

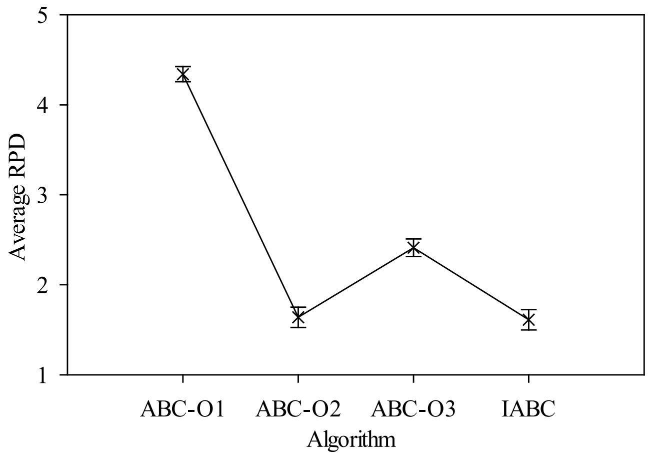


Fig. 8. Means plot and 95% Tukey HSD confidence intervals for ABC algorithms.

All the tested algorithms terminate when the computation time reaches Nt × Nt × 20 milliseconds. After completing the experiments, the obtained results are transferred into RPD values. Afterwards, the analysis of variance (ANOVA) test, which is a powerful statistical technique, is conducted to evaluate the difference. For ANOVA test, the average RPD of all the instances in one run is selected as the response variable, and the algorithm type is regarded as the controlled factors. The ANOVA results show that there is a statistically significant difference among the tested BA algorithms with P-value less than 0.01. And there is also a statistically significant difference among the tested ABC algorithms with P-value less than 0.01. To have a better observation of the difference, Fig. 7 illustrates the means plot of BA algorithms and Fig. 8 presents the mean plot of ABC algorithms. From Fig. 7, it is observed that IBA produces superior performance over the BA-O1, BA-O2 and BA-O3 clearly. From Fig. 8, it is clear that IABC outperforms ABC-O1 and ABC-O3 by a significant margin, demonstrating the effectiveness of the developed improvements. In summary, the comparative studies verify that the developed improvements enhance the proposed IBA and IABC by a large degree. In fact, these improvements are in favour of IABC and IBA, achieving a proper balance between exploitation and exploration.

To have an observation of the convergences of the developed algorithms, Fig. 9 illustrates the best cycle times and the average cycle times of the population under different computation times, where OBA corresponds to OBA-O1, and OABC corresponds to OABC-O1. From this figure, it is observed that IBA show better performance in terms of the average cycle time. And it is observed that IABC has better performance than OABC in terms of the best cycle time and the average cycle time. The OABC shows no clear convergence due to the scout phase whereas the IABC obtains much better convergence. And it is also observed that IBA and IABC employ high-quality solutions to replace the abandoned solutions to preserve the diversity of the population and hence achieve proper balance between exploration and exploitation.

Table 4
The results for small-size instances when $\tau = 100$.

Nt	Ns	CPLEX		IBA		IABC	
		Total C	CPU (s)	Best	Avg	Best	Avg
7	2	482	1	482	482	482	482
	3	325	<1	325	325	325	325
	4	280	<1	280	280	280	280
9	2	569	1.3	569	569	569	569
	3	402	1.3	402	402	402	402
	4	315	1.7	315	315	315	315
11	2	631	1.6	631	631	631	631
	3	443	1.1	443	443	443	443
	4	339	1.6	339	339	339	339
21	2	1366	14.3	1366	1366	1366	1366
	3	923	47.6	923	923	923	923
	4	710	243.8	719	719	719	719
	5	578	543	578	580.5	578	580.1
25	3	1021	35.2	1023	1023	1023	1023
	4	775	83.5	775	775	775	775
	5	626	125.7	626	626.1	626	626.1
	6	542	403.6	542	542.8	542	542.9
35	4	1094	1344.7	1094	1095.9	1094	1095.9
	5	891[a]	+3600	877	881.9	877	879
	6	752[a]	+3600	739	744.2	743	743.2
	7	683[a]	+3600	622	629.1	622	628.4

[a]Integer solution.

On the basis of Salehi, et al. [60], the complexity of the tested algorithms, including BA-O1, IBA, ABC-O1 and IABC are calculated and the total complexity of BA-O1, IBA, ABC-O1 and IABC are $O(N^6)$, $O(N^6)$, $O(N^5)$ and $O(N^7)$; respectively. For space reasons, this section mainly presents the details of calculating the complexity of IBA as an example, where Ni is a number of iterations.

Time complexity of the initial population and its evaluation is $O(s \cdot I^3)$.
Time complexity of sorting the population is $O(s \cdot \log^2 s)$.
Time complexity of updating the best employed bees $O(Ni \cdot e \cdot nep \cdot I^3)$.
Time complexity of updating the remained employed bees $O(Ni \cdot (p - e - 1) \cdot I^3)$.
Time complexity of updating scouts $O(Ni \cdot (S - p - 1) \cdot \mathrm{nn} \cdot I^3 + Ni \cdot \mathrm{e} \cdot \mathrm{nn} \cdot I^3)$, where nn is the number of neighbour solutions for the abandoned solutions.
Total complexity of IBA is $O(N^6)$.

6.3. Experimental results

This section first presents the results for small-size instances by CPLEX solver and algorithms in Table 4. Specifically, the first and second columns of the tables illustrate the number of the tasks (Nt) and the predetermined number of stations (Ns) for each instance; respectively. The third and fourth columns present the achieved total cycle time (Total C) and computational time in seconds (CPU (s)) by the formulated mathematical model. And the

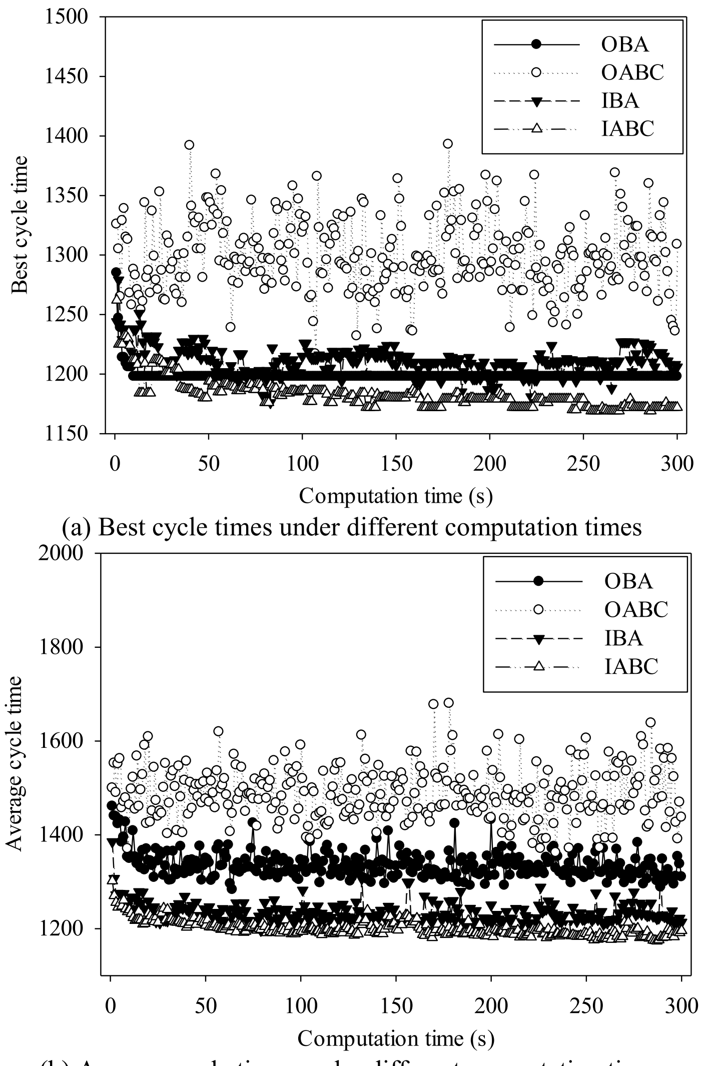


Fig. 9. Best cycle times and the average cycle times under different computation times.

remained columns exhibit the minimum total cycle time (Best) and the average total cycle time (Avg) within 10 independent runs by IBA and IABC when $\tau = 100$.

From this table, it is observed that IBA and IABC obtain optimal solutions for the majority of the small-size instances. Nevertheless, the algorithms demonstrate a clear advantage when solving instances with 35 tasks and more than four stations. For instance, both the best and average results by IBA and IABC outperform the result by CPLEX when solving the instances with 35 tasks and 7 stations. And CPLEX consumes 3600s, whereas the proposed algorithms only consume 122.5 s (35 $\times$ 35 $\times$ 100 ms). This comparative study verifies that the developed model is effective in solving small-size instances, and the proposed algorithms show a clear advantage when solving large-size instances.

To further evaluate the performance of the developed algorithms, this section also presents the results by the tested algorithms under five termination criteria (τ is set to 20, 40, 60, 80 and 100). Table 5 shows the average RPD values in ten runs for all the instances by the implemented algorithms when $\tau = 100$. For space reasons, the detailed results under other termination criteria cannot be provided here, but they are available upon request. From this table, it is clear that IBA and IABC outperform OBA and OABC in terms of the overall RPD; respectively. Specifically, the overall RPD values by OBA and OABC are 3.31 and 3.59, whereas the overall RPD values by IBA and IABC are reduced to 0.94 and 0.84. And the IABC and IBA outperform the GA and PSO for almost all the instances, whereas the two local search algorithms, LAHC and SA, show better performance for a few very large-size instances with 111 tasks and 148 tasks. In short, this comparative study demonstrates that the proposed IABC and IBA obtain competing performance.

To observe the performance of algorithms under different termination criteria, Table 6 illustrates the overall average RPD values by implemented algorithms under five termination criteria. It is observed that all the algorithms obtain better performance with larger computation time. The proposed IBA and IABC produce superior performances over OABC and OBA under all the termination criteria. And IABC is the best performer, and IBA is the second-best performer under all the termination criteria. Again, this comparative study verifies that the IABC and IBA achieve superior performance over the other nine algorithms.

This study also conducts the statistical analysis to check whether the observed difference is statistically significant. Specifically, the multifactor analysis of variance (ANOVA) test is again conducted. For ANOVA test, the average RPD of all the instances in one run is selected as the response variable, and the algorithm type and computation times are selected as the controlled factors. However, an initial ANOVA test demonstrates that the normality

Table 5
The average RPD values for tested instances with $\tau = 100$.

Nt	Ns	LAHC	SA	GA	PSO	DCS	OBA	OABC	ABC1	ABC2	IBA	IABC
7	2	1.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	3	0.00	0.00	0.00	2.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	4	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	2	0.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	3	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	4	1.33	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00
11	2	2.66	0.00	0.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	3	4.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	4	4.75	1.15	0.00	2.51	0.00	0.18	0.00	0.00	0.00	0.00	0.00
21	2	0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
21	3	4.63	0.31	0.00	0.00	0.00	0.31	0.00	0.00	0.00	0.00	0.00
21	4	4.62	2.08	0.76	2.24	1.14	2.10	0.31	0.94	0.89	1.27	1.27
21	5	4.19	2.73	0.00	1.33	0.24	3.77	0.00	0.24	0.00	0.43	0.36
25	3	0.33	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25	4	1.55	0.85	0.00	0.74	0.54	0.00	0.00	0.00	0.00	0.00	0.00
25	5	1.93	1.90	0.00	0.48	0.70	1.52	0.00	0.05	0.00	0.02	0.02
25	6	2.62	2.32	0.18	2.66	0.57	1.11	0.00	0.31	0.30	0.15	0.17
35	4	5.64	2.53	0.00	5.73	0.00	2.07	0.08	0.00	0.00	0.17	0.17
35	5	5.02	2.74	1.14	3.22	1.64	3.12	1.74	0.51	0.62	0.56	0.23
35	6	4.84	3.46	1.99	3.35	2.17	5.28	1.61	1.33	1.11	0.98	0.84
35	7	5.35	5.27	2.33	7.17	3.09	6.29	2.23	1.67	2.19	1.14	1.03
53	5	3.58	2.78	1.30	2.52	1.93	1.99	1.50	1.50	1.35	0.15	0.45
53	6	3.75	4.13	2.62	4.12	2.26	2.98	2.06	1.72	1.85	1.39	1.29
53	7	1.70	0.98	1.48	2.71	1.00	1.87	1.25	0.43	0.30	0.71	0.58
53	8	1.62	1.17	4.76	6.33	2.50	4.96	3.28	0.61	1.15	0.24	0.23
70	7	2.94	2.32	5.51	6.94	3.28	4.65	6.27	0.96	1.23	0.94	0.77
70	8	3.26	3.56	7.29	8.60	4.56	6.32	7.69	1.57	1.44	0.89	0.85
70	9	2.50	2.16	6.52	8.61	4.02	8.15	8.10	0.96	1.00	1.23	0.86
70	10	1.98	3.26	8.21	10.72	4.36	7.33	9.23	1.06	1.20	1.29	1.38
89	8	9.49	2.54	4.35	7.64	3.57	4.80	5.60	1.99	2.74	2.95	4.72
89	9	6.74	1.70	1.99	5.53	4.73	5.58	4.72	1.90	2.87	2.25	3.11
89	10	6.57	2.04	4.51	6.71	4.42	7.22	6.66	2.91	2.74	3.48	2.32
89	11	2.99	2.15	3.30	6.57	3.64	7.72	5.94	2.00	1.82	2.28	1.54
111	9	2.62	2.14	9.95	8.54	5.40	5.87	9.89	2.39	2.28	2.25	1.76
111	10	1.29	1.42	11.26	11.09	5.36	6.97	11.27	2.35	2.60	2.60	1.79
111	11	1.32	1.12	10.81	10.70	4.74	6.91	10.91	1.83	1.76	1.70	1.37
111	12	0.95	0.91	11.26	10.94	5.29	7.81	12.05	2.39	1.85	1.62	1.27
148	10	1.29	0.93	7.19	7.12	3.83	4.35	7.41	1.88	1.64	1.69	1.17
148	11	1.26	0.71	8.20	8.11	3.66	4.24	8.59	2.39	1.87	2.08	1.37
148	12	1.24	0.64	8.60	8.20	4.66	4.40	8.86	2.01	2.03	1.77	1.36
148	13	1.47	1.00	9.89	8.33	5.35	5.61	10.07	2.63	2.41	2.13	2.07
Overall RPD		2.82	1.54	3.31	4.43	2.16	3.31	3.59	0.99	1.01	0.94	**0.84**

Best in bold.

Table 6
Overall RPD values by implemented algorithms.

Algorithm	$\tau = 20$	$\tau = 40$	$\tau = 60$	$\tau = 80$	$\tau = 100$
LAHC	3.18	3.00	2.92	2.87	2.82
SA	1.92	1.70	1.62	1.57	1.54
GA	4.16	3.71	3.50	3.40	3.31
PSO	5.66	5.10	4.81	4.60	4.43
DCS	3.20	2.69	2.45	2.28	2.16
OBA	3.93	3.59	3.45	3.35	3.31
OABC	4.39	3.96	3.81	3.67	3.59
ABC1	1.88	1.44	1.24	1.09	0.99
ABC2	1.94	1.46	1.26	1.10	1.01
IBA	1.64	1.32	1.13	1.04	0.94
IABC	**1.61**	**1.21**	**1.03**	**0.91**	**0.84**

Best in bold.

of the residuals is slightly violated. The non-parametric Friedman test might be of a good choice under this situation, but it cannot analyse the interaction between the algorithm and computation time. Hence, this study conducts both the ANOVA test and the Friedman test, where ANOVA test is conducted to observe the interaction between the algorithm and computation time and the Friedman test is utilized to re-check the results of the ANOVA test. Fig. 10 depicts means plot of the interaction of the two tested factors in ANOVA test, and detailed ANOVA test results and Friedman test results are omitted for space reasons.

From this figure, it is observed that the IABC and IBA obtain better performance than OABC and OBA; respectively. The proposed IABC and IBA are two best performers under all the termination criteria. Also, the proposed IABC and IBA outperform LAHC, GA and PSO by a significant margin. In summary, the computational study demonstrates the superiority of the IABC and IBA over the OABC and OBA, and, as a consequence, the IABC and IBA are capable of solving the considered problem effectively and efficiently.

7. Conclusions and future research

Physical human–robot cooperation in the assembly line provides increasing productivity and flexibility simultaneously. This study presents the first study to tackle the mixed-model assembly line balancing (MMALB) problem with the collaboration between human workers and robots. Firstly, a mixed-integer linear programming (MILP) model is developed to formulate this problem. The computation study shows that this MILP model is capable of achieving the optimal solutions for small-size problems. Secondly, BA and ABC algorithms are improved to tackle large-size problems due to the complexity of the considered problem. The improvements to enhance BA include new employed bee phase to accelerate the evolution of the swarm, and new scout phase to escape from being trapped into local optima and produce high-quality and diverse population. The improvements to enhance

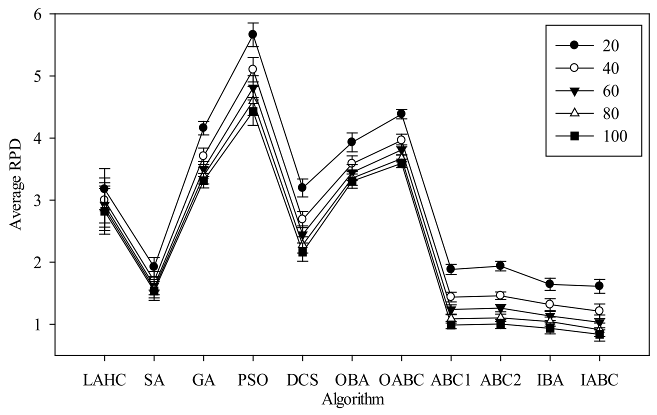


Fig. 10. Means plot and 95% Tukey HSD confidence intervals for the interactions between algorithms and computation times.

ABC include: new onlooker phase to accelerate the evolution of the whole swarm by removing the poor-quality solutions, new scout phase to achieve high-quality solutions while preserving the diversity of the swarm, and local search to enhance exploitation capacity. Thirdly, this study re-implements a set of other nine algorithms to solve the considered problem and conducts a comprehensive study to evaluate the performance of the algorithms. Comparative study between the model and algorithms verifies that the algorithms obtain superior performance when solving large-size problems. Furthermore, a comparative study between eight algorithms demonstrates that the proposed ABC and BA outperform their original versions by a significant margin, and they also achieve competing performance in comparison with nine other algorithms, including the late acceptance hill-climbing algorithm, simulated annealing algorithm, genetic algorithm, particle swarm optimization algorithm, discrete cuckoo search algorithm, the original bee algorithm and three artificial bee colony algorithms..

As far as the managerial implications of the paper are concerned, this study offers a new line design to the manager to increase line efficiency. The proposed model and algorithms help the manager to solve the problem effectively. Future research might develop the branch and bound algorithm to solve both the small-size and large-size problems optimally. And the proposed design should be extended by taking the additional constraints in real applications into account, including positional constraint and zoning constraint. As it is also very important to optimize the assembly cost, it is also suggested to optimize the line efficiency, and the assembly cost simultaneously with the multi-objective optimization method.

CRediT authorship contribution statement

Zeynel Abidin Çil: Methodology, Writing - original draft. **Zixiang Li:** Methodology, Software. **Suleyman Mete:** Writing - original draft, Writing - review & editing. **Eren Özceylan:** Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors are grateful for the anonymous referees for their constructive comments to improve this study. This project is partially supported by National Natural Science Foundation of China under grant 61803287 and China Postdoctoral Science Foundation under grant 2018M642928.

References

[1] S. Akpinar, A. Elmi, T. Bektaş, Combinatorial benders cuts for assembly line balancing problems with setups, European J. Oper. Res. 259 (2017) 527–537.
[2] C.G.S. Sikora, T.C. Lopes, L. Magatão, Traveling worker assembly line (re)balancing problem: Model, reduction techniques, and real case studies, European J. Oper. Res. 259 (2017) 949–971.
[3] R. Ramezanian, A. Ezzatpanah, Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem, Comput. Ind. Eng. 87 (2015) 74–80.
[4] A. Hamzadayi, G. Yildiz, A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints, Comput. Ind. Eng. 62 (2012) 206–215.
[5] D. Romero, P. Bernus, O. Noran, J. Stahre, Å. Fast-Berglund, The Operator 4.0: Human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems, in: Proceedings of IFIP International Conference on Advances in Production Management Systems, 2016, pp. 677–686.
[6] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, J. Alcazar, Gestures for industry: intuitive human-robot communication from human observation, in: Proceedings of the 8th ACM/IEEE International Conference on Human–Robot Interaction, 2013, pp. 349–356.
[7] J. Gao, L. Sun, L. Wang, M. Gen, An efficient approach for type II robotic assembly line balancing problems, Comput. Ind. Eng. 56 (2009) 1065–1080.
[8] Z.A. Çil, S. Mete, K. Ağpak, A goal programming approach for robotic assembly line balancing problem, IFAC-PapersOnLine 49 (2016) 938–942.
[9] Z.A. Çil, S. Mete, E. Özceylan, K. Ağpak, A beam search approach for solving type II robotic parallel assembly line balancing problem, Appl. Soft Comput. 61 (2017) 129–138.
[10] P. Tsarouchi, A.-S. Matthaiakis, S. Makris, G. Chryssolouris, On a human-robot collaboration in an assembly cell, Int. J. Comput. Integr. Manuf. 30 (2017) 580–589.
[11] D. Romero, J. Stahre, T. Wuest, O. Noran, P. Bernus, Å. Fast-Berglund, D. Gorecky, Towards an operator 4.0 typology: A human-centric perspective on the fourth industrial revolution technologies, in: Proceedings of International Conference on Computers & Industrial Engineering, 2016, pp. 1–11.
[12] J. Schmidtler, K. Bengler, robot Collaborations, Fast or accurate? – performance measurements for physical human–robot collaborations, Procedia Manuf. 3 (2015) 1387–1394.
[13] P. Samouei, J. Ashayeri, Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations, Appl. Math. Model. 72 (2019) 259–275.
[14] C. Weckenborg, K. Kieckhäfer, C. Müller, M. Grunewald, T.S. Spengler, Balancing of assembly lines with collaborative robots, Bus. Res. (2019) 1–40.
[15] S. Yaphiar, C. Nugraha, A. Ma'ruf, Mixed model assembly line balancing for human-robot shared tasks, in: International Manufacturing Engineering Conference & The Asia Pacific Conference on Manufacturing Systems 2019, Springer Singapore, Singapore, 2020, pp. 245–252.
[16] N.T. Thomopoulos, Line balancing-sequencing for mixed-model assembly, Manage. Sci. 14 (1967) B–59–B–75.
[17] N.T. Thomopoulos, Mixed model line balancing with smoothed station assignments, Manage. Sci. 16 (1970) 593–603.
[18] H. Gökcen, E. Erel, Binary integer formulation for mixed-model assembly line balancing problem, Comput. Ind. Eng. 34 (1998) 451–461.
[19] E. Erel, H. Gokcen, Shortest-route formulation of mixed-model assembly line balancing problem, European J. Oper. Res. 116 (1999) 194–204.
[20] M. Jin, S.D. Wu, A new heuristic method for mixed model assembly line balancing problem, Comput. Ind. Eng. 44 (2003) 159–169.
[21] I. Kucukkoc, D.Z. Zhang, 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 (2014) 314–333.
[22] I. Kucukkoc, Z. Li, A.D. Karaoglan, D.Z. Zhang, Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm, Int. J. Prod. Econ. 205 (2018) 228–243.
[23] A.S. Simaria, P.M. Vilarinho, A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II, Comput. Ind. Eng. 47 (2004) 391–407.

[24] Z.A. Çil, S. Mete, K. Ağpak, Analysis of the type II robotic mixed-model assembly line balancing problem, Eng. Optim. 49 (2017) 990–1009.
[25] Z. Li, M.N. Janardhanan, Q. Tang, P. Nielsen, Mathematical model and metaheuristics for simultaneous balancing and sequencing of a robotic mixed-model assembly line, Eng. Optim. 50 (2018) 877–893.
[26] U. Saif, Z. Guan, L. Zhang, F. Zhang, B. Wang, J. Mirza, Multi-objective artificial bee colony algorithm for order oriented simultaneous sequencing and balancing of multi-mixed model assembly line, J. Intell. Manuf. 30 (2019) 1195–1220.
[27] Ş. Akpinar, A. Baykasoğlu, Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model, J. Manuf. Syst. 33 (2014) 177–187.
[28] L. Tiacci, Simultaneous balancing and buffer allocation decisions for the design of mixed-model assembly lines with parallel workstations and stochastic task times, Int. J. Prod. Econ. 162 (2015) 201–215.
[29] L. Tiacci, Coupling a genetic algorithm approach and a discrete event simulator to design mixed-model un-paced assembly lines with parallel workstations and stochastic task times, Int. J. Prod. Econ. 159 (2015) 319–333.
[30] M. Aghajani, R. Ghodsi, B. Javadi, Balancing of robotic mixed-model two-sided assembly line with robot setup times, Int. J. Adv. Manuf. Technol. 74 (2014) 1005–1016.
[31] I. Kucukkoc, D.Z. Zhang, Balancing of mixed-model parallel U-shaped assembly lines considering model sequences, Int. J. Prod. Res. 55 (2017) 5958–5975.
[32] I. Kucukkoc, D.Z. Zhang, Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach, Comput. Ind. Eng. 97 (2016) 58–72.
[33] B. Yuan, C. Zhang, X. Shao, Z. Jiang, An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines, Comput. Oper. Res. 53 (2015) 32–41.
[34] P. Chutima, N. Yothaboriban, Multi-objective mixed-model parallel assembly line balancing with a fuzzy adaptive biogeography-based algorithm, Int. J. Ind. Syst. Eng. 26 (2017) 90–132.
[35] F.M. Defersha, F. Mohebalizadehgashti, Simultaneous balancing sequencing and workstation planning for a mixed model manual assembly line using hybrid genetic algorithm, Comput. Ind. Eng. 119 (2018) 370–387.
[36] U. Özcan, H. Çerçioğlu, H. Gökçen, B. Toklu, Balancing and sequencing of parallel mixed-model assembly lines, Int. J. Prod. Res. 48 (2010) 5089–5113.
[37] O. Battaïa, A. Dolgui, A taxonomy of line balancing problems and their solution approaches, Int. J. Prod. Econ. 142 (2013) 259–277.
[38] J. Krüger, T.K. Lien, A. Verl, Cooperation of human and machines in assembly lines, CIRP Ann. 58 (2009) 628–646.
[39] M.P. Mayer, B. Odenthal, M. Faber, C. Winkelholz, C.M. Schlick, Cognitive engineering of automated assembly processes, Hum. Factors Ergon. Manuf. Serv. Ind. 24 (2014) 348–368.
[40] J.T.C. Tan, F. Duan, Y. Zhang, R. Kato, T. Arai, Task modeling approach to enhance man-machine collaboration in cell production, in: IEEE International Conference on Robotics and Automation, 2009, pp. 152–157.
[41] F. Wallhoff, J. Blume, A. Bannat, W. Rösel, C. Lenz, A. Knoll, A skill-based approach towards hybrid assembly, Adv. Eng. Inform. 24 (2010) 329–339.
[42] F. Chen, K. Sekiyama, J. Huang, B. Sun, H. Sasaki, T. Fukuda, An assembly strategy scheduling method for human and robot coordinated cell manufacturing, Int. J. Intell. Comput. Cybern. 4 (2011) 487–510.
[43] S. Takata, T. Hirano, Human and robot allocation method for hybrid assembly systems, CIRP Ann. 60 (2011) 9–12.
[44] J. Rubinovitz, J. Bukchin, E. Lenz, RALB — A heuristic algorithm for design and balancing of robotic assembly lines, CIRP Ann. 42 (1993) 497–500.
[45] Z.A. Çil, S. Mete, E. Özceylan, A mathematical model for semi-robotic assembly line balancing problem: A case study, Int. J. Lean Think. 9 (2018) 70–76.
[46] L.M. Abualigah, A.T. Khader, Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering, J. Supercomput. 73 (2017) 4773–4795.
[47] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, A.H. Gandomi, A novel hybridization strategy for Krill herd algorithm applied to clustering techniques, Appl. Soft Comput. 60 (2017) 423–435.
[48] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, Hybrid clustering analysis using improved Krill herd algorithm, Appl. Intell. 48 (2018) 4047–4071.
[49] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, A combination of objective functions and hybrid Krill herd algorithm for text document clustering analysis, Eng. Appl. Artif. Intell. 73 (2018) 111–125.
[50] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, A new feature selection method to improve the document clustering using particle swarm optimization algorithm, J. Comput. Sci. 25 (2018) 456–466.
[51] L.M.Q. Abualigah, Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering, Springer, 2019.
[52] P. Tapkan, L. Özbakır, A. Baykasoğlu, Bees algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem, Optim. Lett. 6 (2012) 1039–1049.
[53] Q. Tang, Z. Li, L. Zhang, 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 (2016) 146–156.
[54] Q. Pan, An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling, European J. Oper. Res. 250 (2016) 702–714.
[55] Ş. Akpinar, A. Baykasoğlu, Modeling and solving mixed-model assembly line balancing problem with setups. Part II: A multiple colony hybrid bees algorithm, J. Manuf. Syst. 33 (2014) 445–461.
[56] K. Buyukozkan, I. Kucukkoc, S.I. Satoglu, D.Z. Zhang, Lexicographic bottleneck mixed-model assembly line balancing problem: Artificial bee colony and tabu search approaches with optimised parameters, Expert Syst. Appl. 50 (2016) 151–166.
[57] D. Kizilay, M.F. Tasgetiren, O. Bulut, B. Bostan, A discrete artificial bee colony algorithm for the assignment and parallel machine scheduling problem in DYO paint company, in: 2014 IEEE Congress on Evolutionary Computation, CEC, 2014, pp. 653–660.
[58] Q.-K. Pan, M. Fatih Tasgetiren, P.N. Suganthan, T.J. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Inform. Sci. 181 (2011) 2455–2468.
[59] Z. Li, M.N. Janardhanan, Q. Tang, S.G. Ponnambalam, Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times, Swarm Evol. Comput. 50 (2019) 100567.
[60] M. Salehi, H.R. Maleki, S. Niroomand, Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms, Neural Comput. Appl. (2019) http://dx.doi.org/10.1007/s00521-019-04293-8.