Journal of Intelligent Manufacturing (2021) 32:989–1007
https://doi.org/10.1007/s10845-020-01598-7

Cost-oriented robotic assembly line balancing problem with setup times: multi-objective algorithms

Zixiang Li[1,2] · Mukund Nilakantan Janardhanan[3] · S. G. Ponnambalam[4]

Received: 23 November 2019 / Accepted: 23 May 2020 / Published online: 4 June 2020

Abstract
Robots are extensively used during the era of Industry 4.0 to achieve high productivity, better quality and lower cost. While designing a robotic assembly line, production managers are concerned about the cost involved in such a system development. Most of the research reported till date did not consider purchasing cost while optimizing the design of a robotic assembly line. This study presents the first attempt to study the cost-oriented robotic assembly line balancing problem with setup times to minimize the cycle time and total purchasing cost simultaneously. A mixed-integer linear programming model is developed to formulate this problem. The elitist non-dominated sorting genetic algorithm (NSGA-II) and improved multi-objective artificial bee colony (IMABC) algorithm are developed to achieve a set of Pareto solutions for the production managers to utilize for selecting the better design solution. The proposed IMABC develops new employed bee phase and scout phase, which selects one solution in the permanent Pareto archive to replace the abandoned solution, to enhance exploration and exploitation. The comparative study on a set of generated instances demonstrates that the proposed model is capable of achieving the proper tradeoff between line efficiency and purchasing cost, and the proposed NSGA-II and IMABC achieve competing performance in comparison with several other multi-objective algorithms.

Keywords Assembly line balancing · Robotic assembly line · Setup times · Multi-objective optimization · Metaheuristics

Introduction

Assembly process is the last step in the manufacturing of products, where the parts are assembled to achieve the complete products. Assembly lines have been frequently utilized in modern industry to assemble the standardized products due to high productivity (Battaïa and Dolgui 2013). Many practitioners and researchers have paid attentions to the assembly line balancing problem (ALBP) to achieve higher line efficiency. ALBP can be described as allocating a set of tasks to stations with one or several optimization criteria while satisfying the cycle time constraint and precedence constraint. Cycle time constraint requires that the tasks on one station must be completed within the cycle time; precedence constraint demands that the predecessors of one task must be completed before performing this task. Based on the optimization objectives, the ALBP are mainly divided into three categories: type-I ALBP to minimize the number of stations within a given cycle time, type-II ALBP to minimize the cycle time with a given number of stations and type-E ALBP to maximize the line efficiency (Gao et al. 2009).

Due to the increased labor cost and technological progress, there is a clear trend to replace the human workers with robots. The main advantages of robots over the human workers include higher productivity, lower cost, higher assembly quality, continuous working without fatigue and others (Gao et al. 2009; Pereira et al. 2018). The assembly

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

Zixiang Li
lizixiang@wust.edu.cn

S. G. Ponnambalam
ponnambalam.g@vit.ac.in

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

2 Engineering Research Center for Metallurgical Automation and Measurement Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan, China

3 School of Engineering, University of Leicester, Leicester, UK

4 School of Mechanical Engineering, VIT University, Vellore, Tamil Nadu 632014, India

line with robots to operate all the tasks is referred to as robotic assembly line, and the robotic assembly line balancing problem (RALBP) comes up and is needed to be properly tackled and optimized to achieve high line efficiency.

There are many types of robots available to perform the assembly tasks and the robots of different types usually have different purchasing cost. Different robots might need different processing times to perform the same task. The high-tech robots, which are more expensive, might have larger capacities and consume less operation times for most of the tasks; the others, which are cheaper, have limited capacity and might not be capable of performing some tasks. For production manager, the purchasing cost is one of the main factors when designing the robotic assembly line. In some occasions, it is not acceptable to buy the high-tech robots with larger purchasing cost to only pursuit line efficiency. Namely, there is a tradeoff between line efficiency and purchasing cost. In additionn, in robotic assembly line it is necessary to change the tools or remove robotic arms when performing different tasks in real applications (Janardhanan et al. 2019), leading to sequence-dependent setup times. Hence, it is necessary to study the sequence-dependent setup times in the robotic assembly lines.

Among the studies on robotic assembly line, there are two recent and most related papers by Pereira et al. (2018) and Janardhanan et al. (2019). Nevertheless, this study is different from that published in both problem specification and solution method. For instance, the differences between this paper and that by Pereira et al. (2018) are listed as follows. (1) Pereira et al. (2018) considers the cost of performing all the tasks by robots whereas this study considers the purchasing cost; (2) Pereira et al. (2018) develops a single-objective algorithm whereas this study proposes the multi-objective algorithms to achieve a set of Pareto solutions. To the authors' best knowledge, this is the first time to consider the purchasing cost, which is very important for line manager to select the robots within the budget and is ignored in the published studies.

For the aforementioned reasons, this research studies the cost-oriented robotic assembly line balancing problem with setup times to minimize the cycle time and total purchasing cost with multi-objective algorithms. This study presents two contributions as follows. (1) This study considers the purchasing cost for the first time and develops a mixed-integer linear programming (MILP) model to formulate this problem. The formulated MILP model is capable of solving the small-size problems optimally when optimizing either objective or combining the two objectives into one weighted objective. (2) As the two objectives are conflicted in many occasions observed in preliminary experiments, this study presents two population-based multi-objective algorithms to tackle this problem: the elitist non-dominated sorting genetic algorithm (NSGA-II) and improved multi-objective artificial bee colony (IMABC) algorithm. The proposed IMABC utilizes crossover operator in the employed bee phase to emphasize exploration and utilize new scout phase, which selects one solution in the permanent Pareto archive to replace the abandoned solution, to emphasize exploitation. This study also conducts a comprehensive study to evaluate the performance of the proposed algorithms. Computational study demonstrates that the proposed NSGA-II and IMABC produce competing performance in comparison with the multi-objective simulated annealing algorithm and the original multi-objective artificial bee colony (OMABC) algorithm. And the IMABC produce slightly better performance than NSGA-II when the computation time is large.

The remainder of this paper is structured as follow. "Literature review" section reviews the related and recent literature and "Mathematical formulation" section formulates the considered problem. Subsequently, the developed multi-objective methodologies are detailed in "Proposed multi-objective algorithms" section and an illustrated example is provided in "An illustrated example" section to highlight the features of the studied problem. The computational study is conducted in "Experimental tests and results" section to evaluate the developed algorithms. Finally, the conclusions and future research avenues are provided in "Extension of the model and algorithms for type-I RALBP" section.

Literature review

As there is no published study considering all the features of the studied problem, this section mainly reviews the papers on three related topics: RALBP, ALBP with setup times and cost-oriented ALBP.

Since the pioneering work by Rubinovitz and Bukchin (1991), many studies have been published to enrich the solution methods and extend the RALBP. The applied methods might be divided into three categories: exact methods, heuristic methods and metaheuristic methods. The exact method includes the branch and bound method by Rubinovitz et al. (1993) to minimize the number of stations, branch, bound and remember algorithm by Borba et al. (2018) to reduce cycle time. The heuristics and metaheuristics algorithms that has been utilized to solve different types of robotic assembly line balancing problems include strong cutting plane algorithm (Kim and Park 1995), beam search heuristic (Çil et al. 2017b), genetic algorithms (Levitin et al. 2006; Gao et al. 2009), hybrid algorithms (Daoud et al. 2014), particle swarm optimization (Nilakantan et al. 2015a), bio-inspired algorithms (Nilakantan et al. 2015b), differential evolution algorithm (Nilakantan et al. 2017b), simulated annealing algorithm (Li et al. 2018b), migrating birds optimization algorithm (Janardhanan et al. 2019) and others. Regarding the extensions of the RALBP, many studies consider more realistic objectives. Yoosefelahi et al. (2012) study the multiple objective with three multi-objective evolutionary strategies. Nilakantan et al. (2015a, 2016) investigate the energy consumption, Nilakantan et al. (2017b) and Pereira et al. (2018) study the cost of performing all the tasks by robots, Nilakantan et al. (2017a) investigate the carbon footprint, and Zhou and Wu (2019) minimizes the number of station and the area of each station. And there are also some studies to consider different line configurations and different products (models). Specifically, the RALBP is extended to other line layouts including U-shaped line (Nilakantan and Ponnambalam 2016; Li et al. 2019a; Zhang et al. 2018, 2019), two-sided line(Li et al. 2016, b, 2018a, b; Aghajani et al. 2014) and parallel line (Çil et al. 2017b). Çil et al. (2017a), Rabbani et al. (2016) and Aghajani et al. (2014) consider the mixed-model RALBP, and Li et al. (2018c) tackle the balancing and sequencing of mixed-model robotic assembly line.

Although many papers neglect the setup times or include them into the task operation times, the setup times cannot be neglected in many occasions and they should be treated separately from task operation times to improve resource utilization in modern competitive industry context. Andrés et al. (2008) provide the first study to consider the sequence-dependent setup times and present a heuristic method to tackle this problem. Scholl et al. (2008) consider the sequence-dependent task time increments and Scholl et al. (2013) extend the model in Andrés et al. (2008), where the sequence-dependent setup times are divided into forward setup times and backward setup times. After that, Seyed-Alagheband et al. (2011) develop a simulated annealing algorithm to tackle the ALBP with sequence-dependent setup times to minimize the cycle time. Yolmeh and Kianfar (2012) tackle the ALBP with sequence-dependent setup times utilizing a hybrid genetic algorithm. Hamta et al. (2013) tackle the multi-objective ALBP with flexible operation times, sequence-dependent setup times and learning effect utilizing a hybrid swarm optimization algorithm. There are more literatures reported on sequence-dependent setup times on other assembly line configuration including mixed-model assembly line(Akpinar and Baykasoğlu 2014a, b; Akpınar et al. 2013), robotic assembly line (Janardhanan et al. 2019), U-shaped assembly line (Şahin and Kellegöz 2017), two-sided assembly line (Özcan and Toklu 2010; Delice 2019), parallel assembly lines (Özcan 2019), robotic two-sided assembly line (Aghajani et al. 2014; Li et al. 2019b) and others.

Regarding the cost-oriented ALBP, Amen (2000) surveys the heuristic methods for cost-oriented ALBP and Amen (2001) evaluates these heuristic methods. Later, Amen (2006) formulates the cost-oriented ALBP, where the cost of performing tasks is considered. After that, more methods are developed for cost-oriented ALBP (Padrón et al. 2009; Salehi et al. 2019) and the cost-oriented ALBP is extended to consider other assembly line configuration (Roshani et al. 2012; Nilakantan et al. 2017b; Foroughi and Gökçen 2019; Pereira et al. 2018), to cite just a few. Among these studies, there are two papers that reported on the cost-oriented robotic assembly lines by Nilakantan et al. (2017b) and Pereira et al. (2018). In Nilakantan et al. (2017b), it is based on the cost of performing a task by one robot and this study minimizes the total cost of performing all the tasks. Pereira et al. (2018) extend the model in Nilakantan et al. (2017b) by considering the fixed cost of the robots and this paper minimizes the sum of fixed cost and variable cost (total cost of performing all the tasks).

From the aforementioned literature review, although there are two studies considering the cost of performing the tasks on the robotic assembly line, there is no study considering the purchasing cost of robots, which cannot be neglected for production manager to design robotic assembly line. Hence, this study fulfills the gap by studying the cost-oriented robotic assembly line balancing problem with setup times to minimize the cycle time and purchasing cost simultaneously.

Mathematical formulation

This section first presents the problem description and later details the formulated mathematical model.

Problem description

For the considered problem, there are a set of candidate robot types. The robots of different types have different characteristics and they might have different operation times when performing the same task. The RALBP has two interrelated sub-problems: (1) task assignment; (2) robot selection and allocation. For task assignment, the tasks are divided and allocated to a set of stations, where the predecessors of one task must be assigned to the former or the same station. Regarding robot selection and allocation, for each station, one robot is selected from several candidate robot types and is allocated to this station to perform a set of tasks.

Figure 1 illustrates an example layout of robotic assembly line with four stations. Here, there are three types of robots (the robots of the same type are allocated to the former two stations) and four robots on the four stations. All the robots perform the tasks on the corresponding station to achieve complete products. The quality of the robotic line depends on both the task assignment and robot selection and allocation.

On the basis of Gao et al. (2009) and Janardhanan et al. (2019), the main model assumptions are provided here. On the basis of the observations of the robots in assembly lines, it is set that the expensive robots have larger capacities and

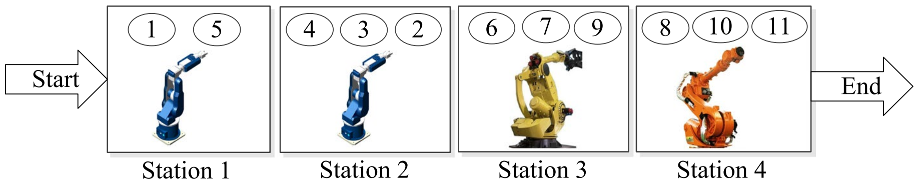


Fig. 1 Layout of robotic assembly line

consume less operation times for most tasks in this study. Robots that are less expensive might not be capable of performing some tasks and consume more operation times for most tasks, but the less expensive robot might need less operation times for a small portion of tasks.

(1) A single product is assembled on the straight robotic assembly line.
(2) Only forward sequence-dependent setup times are considered following Aghajani et al. (2014) and Janardhanan et al. (2019).
(3) The task operation times, setup times and precedence diagram are known and fixed.
(4) The task operation times and setup times depend on the robot types.
(5) One robot is purchased from a set of candidate robot types for each station and the robots of the same type can be allocated to several stations.
(6) Each robot type has a specific purchasing cost and the number of robot types is greater than zero.
(7) The robots of each type are available without limitation.
(8) The maintenance operations, material handling and loading and unloading times are negligible.

Mathematical model

For clarification, the utilized notations are first presented as follows.

Indices	
i, j	Task index, $i,j \in \{1, 2, \ldots, nt\}$, where nt is the number of tasks
k	Station index, $k \in \{1, 2, \ldots, ns\}$, where ns is the number of stations
p	Position index inside the schedule of a station
r	Robot index, $r \in \{1, 2, \ldots, nr\}$, where nr is the available robot types
Parameters	
I	Set of tasks, $I = \{1, 2, \ldots, nt\}$
K	Set of stations, $K = \{1, 2, \ldots, ns\}$
R	Set of robot types, $R = \{1, 2, \ldots, nr\}$
t_{ir}	Operation time of task i by robot r
PT_i	Set of tasks that precede task i
Nm_k	Maximum number of tasks on station k
N^{max}	Maximum number of tasks on any station, $N_{max} = \max_k \{Nm_k\}$
s_{ijr}	Setup time when task j is performed just after task i and they are on the same station and are operated by robot r
$\wp$	Set of pairs of tasks (i,j) in which task i is the immediate predecessor of task j
c_r	Purchasing cost of robot r
ψ	A very large positive number
UB_{CT}	The upper bound of cycle time provide by the line manager; UB_{CT} is set to a very large positive number if there is no limit
Decision variables	
CT	Cycle time
x_{irkp}	1, if task i is operated by robot r and in position p in the sequence of tasks assigned to station k; 0, otherwise
y_{irk}	1, if task i is operated by robot r and in the last position in the sequence of tasks assigned to station k; 0, otherwise
z_{ijrk}	1, if task i and task j are operated by robot r and task i is performed immediately before task j on station k in the same or in the next cycle; 0, otherwise
w_{rk}	1, if robot r is allocated to station k; 0, otherwise

On the basis of Janardhanan et al. (2019), the formulated model is expressed with expressions (1–14) to minimize the cycle time and total purchasing cost simultaneously. Here, the objective function (1) optimizes the cycle time and objective function (2) optimizes the total purchasing cost. Constraint (3) deals with the assignment of tasks, indicating that each task must be assigned to one position inside one station and is operated by one robot. Constraints (4) and (5) deal with the robot allocation. Constraint (4) ensures that each station is allocated with one robot and constraint (5) ensures that the tasks on one station must be operated by the same robot. Constraints (6) and (7) tackle the task assignment inside one station, indicating that there should be at most one task in each position of each station and the tasks should be assigned in increasing positions in the schedule of one station. Constraint (8) handles the precedence constraint, where the predecessors should be allocated to the former station or the former position of the same station. Constraint (9) is the cycle time constraint, ensuring that the total operation time should be less than or equal to the cycle time. Constraints (10)–(12) together determine the value of

z_{ijrk}. Constraint (10) ensures that z_{ijrk} is equal to 1 when task i is performed immediately before task j on station k in the same cycle. And constraints (11) and (12) indicate that z_{ijrk} is equal to 1 when task i is performed immediately before task j on station k in the next cycle. Constraint (13) indicates that the cycle time must be less than or equal to the upper bound of the cycle time provide by the line manager. Constraint (14) restricts the values of the decision variables.

$$\text{Min}\, f_1 = CT \tag{1}$$

$$\text{Min}\, f_2 = \sum_{r\in R}\sum_{k\in K} c_r w_{rk} \tag{2}$$

$$\sum_{r\in R}\sum_{k\in K}\sum_{p=1}^{Nm_k} \mathrm{x}_{irkp} = 1 \quad \forall i \in \mathrm{I} \tag{3}$$

$$\sum_{r\in R} w_{rk} = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{i\in I} x_{\mathrm{irkp}} \le \psi \cdot w_{rk}\ \forall r \in R, k \in K, \quad p = 1, \ldots, Nm_k \tag{5}$$

$$\sum_{i\in I} x_{irkp} \le 1\ \forall r \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \quad \mathrm{p} = 1, \ldots, Nm_k \tag{6}$$

$$\sum_{i\in I} \mathrm{x}_{\mathrm{irk,p+1}} \le \sum_{i\in I} \mathrm{x}_{\mathrm{irkp}}\ \forall \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \quad \mathrm{p} = 1, \ldots, Nm_k - 1 \tag{7}$$

$$\begin{aligned}&\sum_{r\in R}\sum_{k\in K}\sum_{p=1}^{Nm_k} (N^{max} \cdot (k-1) + p) \cdot \mathrm{x}_{\mathrm{irkp}} \\ &\le \sum_{r\in R}\sum_{k\in K}\sum_{p=1}^{Nm_k} (N^{max} \cdot (k-1) + p) \\ &\cdot \mathrm{x}_{\mathrm{jrkp}} \quad \forall (i, \mathrm{j}) \in \wp\end{aligned} \tag{8}$$

$$\sum_{i\in I}\sum_{r\in R}\sum_{p=1}^{Nm_k} t_{ir} \cdot \mathrm{x}_{\mathrm{irkp}} + \sum_{i\in I}\sum_{\mathrm{j}\in \mathrm{I}\wedge(i\ne j)}\sum_{r\in R} s_{ijr} \cdot \mathrm{z}_{\mathrm{ijrk}} \le CT \quad \forall k \in K \tag{9}$$

$$\begin{aligned}&\mathrm{x}_{\mathrm{irkp}} + \mathrm{x}_{\mathrm{jrk,p+1}} \le 1 + z_{ijrk}\ \forall \mathrm{i,j} \in \mathrm{I} \\ &\wedge (\mathrm{i} \ne \mathrm{j}) \wedge (\mathrm{j} \notin PT_i), \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \\ &\mathrm{p} = 1, \ldots, Nm_k - 1\end{aligned} \tag{10}$$

$$\begin{aligned}&\mathrm{x}_{\mathrm{irkp}} - \sum_{\forall \mathrm{j}\in \mathrm{I}|(i\ne j)\wedge(\mathrm{j}\notin PT_i)} \mathrm{x}_{\mathrm{jrk,p+1}} \le y_{irk} \\ &\forall \mathrm{i} \in \mathrm{I}, \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \quad \mathrm{p} = 1, \ldots, Nm_k - 1\end{aligned} \tag{11}$$

$$y_{irk} + \mathrm{x}_{\mathrm{jrk1}} \le 1 + z_{ijrk}\ \forall \mathrm{i,j} \in \mathrm{I} \wedge (\mathrm{i} \ne \mathrm{j}) \wedge (\mathrm{i} \notin PT_j), \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K} \tag{12}$$

$$CT \le UB_{CT} \tag{13}$$

$$x_{irkp}, y_{irk}, z_{ijrk}, w_{rk} \in \{0, 1\} \tag{14}$$

It is to be noted that this formulated model is a MILP model. This model is capable of solving the small-size problems optimally when optimizing either objective or combining the two objectives into one weighted objective utilizing expression (15). In expression (15), w_1 and w_2 are two weights and f_1^{min} and f_1^{max} (f_2^{min} and f_2^{max}) are minimum value and the maximum value of f_1 (f_2). For multi-objective optimization algorithms, the w_1 and w_2 should be tested in many combinations to obtain a set of Pareto solutions.

$$\begin{aligned}\text{Min}\, f = &\ w_1 \cdot \left(f_1 - f_1^{min}\right) / \left(f_1^{max} - f_1^{min}\right) \\ &+ w_2 \cdot \left(f_2 - f_2^{min}\right) / \left(f_2^{max} - f_2^{min}\right)\end{aligned} \tag{15}$$

Proposed multi-objective algorithms

This section presents two multi-objective algorithms, NSGA-II and IMABC, to tackle the studied problem. NSGA-II is selected as it has been widely applied in solving the multi-objective optimization problems and achieves promising performance. IMABC is selected due to its successful applications in RALBP (Zhang et al. 2018) and other scheduling problems (Saif et al. 2019). Compared with the OABC, the proposed IMABC utilizes two improvements: (1) IMABC utilizes crossover operator in the employed bee phase to emphasize exploration; (2) IMABC utilize new scout phase, which selects one solution from the permanent Pareto archive to replace the abandoned solution, to emphasize exploitation.

Encoding and decoding

As the considered problem contains two interrelated subproblems, this study utilizes two encoding vectors: task permutation vector and robot selection vector. Task permutation vector corresponds to one sequence of the tasks (see Fig. 2), where the former tasks in the task permutation should be allocated at first. Robot selection vector determines the selected robot for each station, where the code r in position k donates that robot r is allocated to station k.

As the model in "Mathematical model" section minimizes the cycle time, an initial cycle time or CT is essential in the decoding process. The main decoding procedure is provided in Algorithm 1, where this initial cycle time CT is updated with $CT \leftarrow CT + 1$ until all the tasks can be completed within the initial cycle time. Here, the robot selection can be

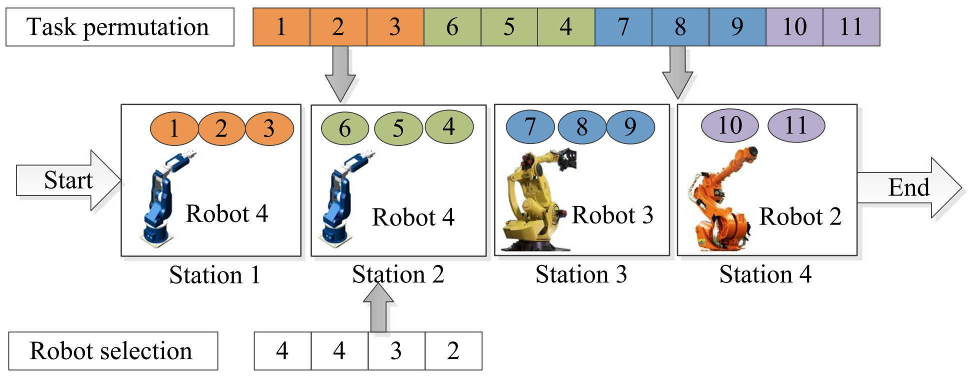


Fig. 2 Illustrated solution presentation

determined directly by the robot selection vector, whereas the detailed task assignment is determined during the decoding procedure. Firstly, an initial cycle time CT is set as $CT \leftarrow 2 \cdot CT_{LB}$ in initialization or $CT \leftarrow CT_{Best} - 1$ during evolution, where CT_{LB}, $CT_{LB} = \left(\sum_i \min_r t_{ir}\right)/ns$, is the lower bound of cycle time and CT_{Best} is the best cycle time during the evolution. Subsequently, the tasks are assigned to stations within this initial cycle time. One station is assigned with tasks as more as possible, where a task is assignable when all its predecessors have been assigned and it can be completed within the given CT when it is on one of the former $ns - 1$ stations. If one station cannot endure more task, the next station is opened to endure the remained tasks. Notice that the cycle time constraint is disabled when assigning the tasks to the last station to achieve a complete solution. Finally, whether all the tasks could be completed within this cycle time is checked. Update the CT with $CT \leftarrow CT + 1$ when the completion time of the last station is larger than CT; terminate this procedure and calculate the objective values, otherwise.

Algorithm 1: Decoding procedure

```
Determine the robot selection;
Set $CT$ as the initial cycle time;
While (1)
  For $k := 1$ to $ns$
    While (1)
      Obtain the assignable task set;
      % A task is assignable when 1) all its predecessors have been assigned; 2) it can
      be completed within the given $CT$ when it is on one of the former $ns - 1$
      stations.
      If (assignable task set is empty)
        Terminate this while loop;
      Endif
      Select the assignable task which is in the former position of the task permutation
      vector;
      Assign the selected task to the current station and update the remained capacity;
    Endwhile
  Endfor
  If (the completion time of last station is not larger than $CT$)
    Terminate this procedure and calculate the objective values;
  Else $CT \leftarrow CT + 1$
  Endif
Endwhile
```

An example of the encoding and decoding procedure is provided in Fig. 2. Here, the robot selection vector provides the robot allocation directly and the detailed task assignment is determined during the decoding procedure. As you can see, the tasks in the former positions of the task permutations are allocated to the former stations.

NSGA-II algorithm

Based on Deb et al. (2002), the main procedure of NSGA-II is illustrated in Algorithm 2, where *PS* is the number of population size. After initializing the parent population, the promising individuals with the smaller ranks or lager crowding distances when the ranks are the same are selected as the new offspring population utilizing the binary tournament selection. The crossover operator and mutation operator are applied in sequence to modify the offspring population. Afterwards, the elitism is introduced to select the best *PS* individuals from the parent population and the offspring population as the new parent population. The main loop is repeated until the termination criterion is met and the outcome of this multi-objective algorithm is a set of Pareto solutions.

Algorithm 2: Main procedure of the NSGA-II algorithm

```
Initialize the parent population $(s_1, s_2, \cdots, s_{PS})$ randomly.
While (Termination criterion is not satisfied)
    Achieve the ranks and the crowding distances of the parent population;
    Obtain the offspring population $(s_1', s_2', \cdots, s_{PS}')$ utilizing the binary tournament
    selection (the individual with smaller rank or lager crowding distance is
    selected);
    Apply the crossover operator and mutation operator to modify the offspring
    population;
    Achieve the ranks and the crowding distances of the two populations;
    Select the best $PS$ individuals from the two populations (with smaller ranks or
    lager crowding distances when the ranks are the same) as the new parent
    population;
Endwhile
```

The calculation of the ranks and the crowding distances is the vital segment in the NSGA-II and it is clarified as follows. For the considered problem, a solution s_1 dominates another solution s_2, i.e., $s_1 \prec s_2$, when either objective of solution s_1 is worse than that of solution s_2. The ranks of the solutions are calculated as follows utilizing the non-dominated sort method (Deb et al. 2002). Firstly, the non-dominated solutions are removed out from the original population and added to nondominated set F_1 with the rank of 1. Subsequently, the new non-dominated solutions are removed out from the population and added to nondominated set F_2 with the rank of 2. This procedure terminates until the population is empty. Regarding the crowding distance, it is calculated utilizing crowding distance assignment (Deb et al. 2002). For the solutions in $F(F = [F[1], \ldots, F[l]], l = |F|)$ with the same rank, the solutions are sorted in the increasing order of the f_1. Afterward, the distances of the boundary positions are set as a very large positive number, $F[1]_{distance} = F[l]_{distance} = \infty$. And for the remained individual i $(2 \le i \le l-1)$, the crowding distance is calculated with $F[i]_{distance} = \left(F[i+1] \cdot f_1 - F[i-1] \cdot f_1\right) / \left(f_1^{max} - f_1^{min}\right) + \left(F[i-1] \cdot f_2 - F[i+1] \cdot f_2\right) / \left(f_2^{max} - f_2^{min}\right)$. Detailed descriptions of the calculating the ranks and the crowding distances refer to Deb et al. (2002).

OMABC algorithm

The main procedure of the original multi-objective artificial bee colony (OMABC) algorithm by Saif et al. (2014), is illustrated in Algorithm 3. The OMABC has two parameters: the number of swarm (*PS*) and the number of (*limit*) trials after which a food source should be abandoned. OMABC begins with initializing the swarm $(s_1, s_2, \ldots, s_{PS})$ randomly. Afterwards the main loop, comprising of the employed bee phase, onlooker phase and scout phase, is repeated until the termination criterion is met. In the employed bee phase, the new swarm is achieved with neighbor operator and the ranks and the crowding distances of the two swarm are calculated utilizing the non-dominated sort method and crowding distance assignment (Deb et al. 2002). Then, the incumbent individual is replaced with the new individual when the better performance is achieved. In the onlooker phase, one individual is selected with the binary tournament selection and one neighbor solution is obtained with the neighbor operator. Once the new swarm is achieved, the best *PS* individuals with smaller ranks or lager crowding distances are selected as the new swarm. In the scout phase, the individual, which has not been improved for consecutive *limit* times is replaced with a randomly generated solution.

Improved multi-objective artificial bee colony (IMABC) algorithm

To enhance the exploration and exploitation capacity, this study proposes the IMABC algorithm and the main procedure is presented in Algorithm 4. The main procedure of IMABC is similar to the OMABC, whereas IMABC proposes two main modifications. (1) In the employed bee phase, the new swarm is obtained utilizing the crossover operator and this modification aims at expanding the search space to emphasize exploration. (2) The proposed IMABC records all the non-dominated solutions in the permanent Pareto archive and IMABC replaces the abandoned individual with a solution in the permanent Pareto archive rather than a randomly generated solution. This modification guarantees the quality of the scouts and hence emphasizes exploitation capacity.

Algorithm 3: Main procedure of the OMABC algorithm

```
Initialize the swarm $(s_1, s_2, \cdots, s_{PS})$ randomly;
While (Termination criterion is not satisfied)
    % Employed bee phase
    Obtain the new swarm $(s_1', s_2', \cdots, s_{PS}')$ with neighbor operator;
    Achieve the ranks and the crowding distances of the two swarm;
    For $p$:=1 to $PS$ do
        Replace $s_p$ with $s_p'$ when $s_p'$ has smaller rank or larger crowding distance
        when $s_p$ and $s_p'$ have the same rank;
    Endfor
    % Onlooker phase
    For $p$:=1 to $PS$ do
        Select one individual with the binary tournament selection and obtain a new
        individual $s_p'$ with neighbor operator;
    Endfor
    Achieve the ranks and the crowding distances of the two swarms;
    Select the best $PS$ individuals (with the smaller ranks or lager crowding distances
    when the ranks are the same) as the new swarm;
    % Scout phase
    Replace the individual, which has not been updated for consecutive limit times,
    with a randomly generated solution;
Endwhile
```

Algorithm 4: Main procedure of the IMABC algorithm

```
Initialize the swarm $(s_1, s_2, \cdots, s_{PS})$ randomly;
While (Termination criterion is not satisfied)
   % Employed bee phase
   Obtain the new swarm $(s_1', s_2', \cdots, s_{PS}')$ utilizing crossover operator;
   % $s_1'$ is obtained by combining $s_1$ and another different individual with crossover operator.
   Achieve the ranks and the crowding distances of the two swarm;
   For $p$:=1 to $PS$ do
      Replace $s_p$ with $s_p'$ when $s_p'$ has smaller rank or larger crowding distance when $s_p$ and $s_p'$ have the same rank;
   Endfor
   % Onlooker phase
   For $p$:=1 to $PS$ do
      Select one individual with the binary tournament selection and obtain a new individual $s_p'$ with neighbor operator;
   Endfor
   Achieve the ranks and the crowding distances of the two swarms;
   Select the best $PS$ individuals (with the smaller ranks or lager crowding distances when the ranks are the same) as the new swarm;
   % Scout phase
   Replace the individual, which has not been improved for consecutive limit times, with one solution in the permanent Pareto archive randomly;
Endwhile
```

Neighbor structure

Both the NSGA-II and IMABC utilize swap operator and insert operator as the neighbor operator for modifying the task permutation vector; they propose swap operator and alternation operator as the neighbor operator for modifying the robot selection vector. Specifically, for swap operator to modify the task permutation, two different tasks are randomly selected and their positions in the task permutation are exchanged. For the insert operator to modify the task permutation, one task is randomly selected and it is inserted into a different position in the task permutation randomly. For swap operator to modify the robot selection vector, two different positions are randomly selected and the robots in these two positions of the robot selection vector are exchanged. For the alternation operator to modify the robot selection vector, one position is randomly selected and the robot in this position of the robot selection vector is replaced with a different robot. When applying these neighbor operators, the task permutation or robot selection vector is randomly selected and later one of the two neighbor operators is randomly selected to modify the corresponding vector.

To combine two individuals, the NSGA-II and IMABC utilize two-point crossover operator as the crossover operator. Figure 3 provides an example of applying the two-point crossover operator to modify the task permutation vector.

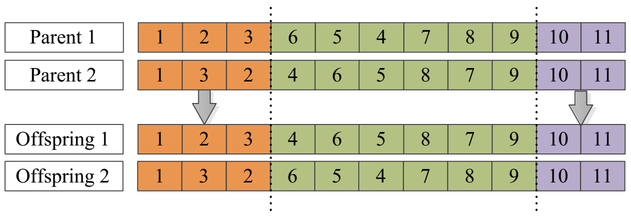


Fig. 3 Two-point crossover operator to modify the task permutation vector

Table 1 Precedence relations and task operation times

Task	Successors	Task operation times			
		Robot 1	Robot 2	Robot 3	Robot 4
1	2, 3, 4, 5	186	61	69	56
2	6	250	166	122	48
3	7	149	132	52	60
4	7	117	67	124	46
5	7	211	59	45	29
6	8	176	107	113	82
7	9	117	84	54	56
8	10	115	69	46	51
9	11	98	125	56	38
10	11	103	76	56	89
11	–	174	62	113	100

An illustrated example

This section illustrates an example to present the main features of the studied problem. This instance has 11 tasks, 4 stations and 4 types of available robots. The purchasing costs of the four types of robots are set as 4.37, 6.08, 7.36 and 8.67; respectively. Table 1 exhibits the precedence relations and the task operation times by robots and Table 2 illustrates the sequence-dependent setup times between tasks by robots.

Table 3 presents the detailed task assignment and robot allocation with an initial cycle time of 170. Here, the second row and the third row present the utilized encoding scheme. The remaining rows present the achieved task assignment and robot allocation. For one station, the total operation time on one station is sum of the operation times of tasks and setup times. For instance, the operation times of task 1, task 2 and task 3 on station 1 by robot 4 is 56, 48 and 60. The setup time between task 1 and task 2, setup time between task 2 and task 3 and setup time between task 3 and task 1 are 5, 0 and 1; respectively. Hence, the completion time of station 1 is calculated as $56+48+60+5+0+1=170$. Afterwards, the cycle time is the maximum value of the completion times of all stations and the total purchasing cost is the sum of the purchasing costs of the robots on all the stations.

Table 2 The sequence-dependent setup times

Robot 1											Robot 2										
0	5	21	23	23	0	9	16	0	11	16	0	3	2	3	7	12	13	10	13	5	5
18	0	16	2	23	7	2	11	5	5	7	8	0	2	3	2	3	8	13	8	5	0
14	14	0	7	21	2	9	2	0	0	23	7	8	0	5	10	2	7	3	2	2	2
23	14	7	0	0	7	5	0	18	5	21	2	8	2	0	12	2	13	2	3	8	12
14	14	14	7	0	9	9	18	5	0	16	8	3	5	7	0	7	12	8	10	5	7
18	21	11	2	23	0	16	0	0	11	7	13	13	5	2	7	0	5	7	7	7	5
14	16	9	14	11	16	0	16	0	21	16	2	12	12	2	7	10	0	0	3	10	2
18	5	14	11	2	2	14	0	16	11	16	8	10	8	10	3	5	3	0	2	7	5
16	11	5	14	14	9	18	2	0	23	14	0	2	10	5	13	0	10	7	0	10	0
18	18	18	18	0	7	14	9	9	0	11	10	12	0	2	8	10	3	0	12	0	3
18	11	23	9	2	5	23	18	2	9	0	0	13	5	5	3	13	10	8	5	3	0
Robot 3											Robot 4										
0	0	7	1	4	10	7	7	4	10	7	0	5	0	5	5	5	0	0	0	5	0
4	0	8	1	10	0	3	5	1	4	0	6	0	0	1	5	6	5	2	3	6	0
4	10	0	5	10	3	4	0	11	5	1	1	3	0	0	2	0	0	2	6	3	1
1	0	3	0	7	7	10	3	3	5	8	5	3	2	0	0	0	1	1	2	6	3
0	8	11	3	0	7	0	5	3	4	7	1	0	3	3	0	5	6	6	3	0	2
1	4	5	1	0	0	0	7	3	8	4	5	3	2	0	5	0	6	6	7	1	5
0	10	10	8	5	1	0	5	4	7	7	2	6	5	6	6	2	0	3	0	6	5
4	1	7	1	10	10	10	0	7	5	1	1	5	2	6	1	7	0	0	0	7	6
11	1	7	1	5	8	1	1	0	3	0	0	3	2	6	2	5	0	3	0	3	6
8	5	4	3	8	8	7	5	5	0	3	5	3	1	3	0	5	1	5	3	0	2
1	0	10	10	8	3	0	7	1	1	0	5	6	0	0	2	5	3	0	5	2	0

Table 3 Detailed task assignment and robot allocation

	Station 1	Station 2	Station 3	Station 4
Encoding				
Task permutation	1, 2, 3, 6, 5, 4, 7, 8, 9, 10, 11			
Robot selection	4	4	3	2
Decoding				
Task assignment	1, 2, 3	6, 5, 4	7, 8, 9	10, 11
Robot selection	4	4	3	2
Operation times of tasks	56, 48, 60	82,29,46	54,46,56	76,62
Setup times	5, 0, 1	5, 3, 0	5, 7, 1	3, 3
Completion time	170	165	169	144
Cycle time	170			
Purchasing cost	8.67	8.67	7.36	6.08
Total purchasing cost	8.67 + 8.67 + 7.36 +6.08 = 30.78			

Experimental tests and results

This section tests the performance of the developed model and algorithms. The experimental setting is first presented in "Experimental setting" section to clarify the utilized instances. "Evaluation indicators" section presents the evaluation indicators and "Model evaluation" section tests the formulated model. Finally, "Comparative study" section conducts the comparative study to further test the developed algorithms.

Experimental setting

As there are no reported literature on the considered problem, it is necessary to generate a set of test instances. Since this study considers the setup times, first the 33 instances with lower setup times from Janardhanan et al. (2019) is selected as the base. However, these instances do not have purchasing cost data and this study presents the following method to modify the instances. Firstly, for each instance it is set that at most four available types of robots are available and the operation times and setup times of the former four robots are selected. Then, for each type of robot, a random number (γ) within [2, 10] is selected as the purchasing cost and the task operation times and setup times by this robot are replaced with $t_{ir} \leftarrow \left(10 \cdot t_{ir}\right)/\gamma$ and $s_{ijr} \leftarrow \left(10 \cdot s_{ijr}\right)/\gamma$. It is to be noted that in the published instances, these robots have different operation times for the same task. Hence, for most instances the task operation times by the robot with higher purchasing cost is shorter than that by the robots with lower purchasing cost, whereas the task operation times by the robot with higher purchasing cost might be larger than that by the robots with lower purchasing cost for a small portion of tasks. After modification, there are a set of 33 instances with nine precedence graphs, ranging from the small-sized instances with 7 tasks to the large-sized instances with 297 tasks. The detailed precedence relation and task operation times of the tested instances are available upon request.

To evaluate the performance of developed NSGA-II and IMABC, this study also implemented another multi-objective algorithm named multi-objective simulated annealing algorithm (MSA) reported in Li et al. (2016b). MSA is a local search method and it achieves promising performance in robotic two-sided assembly line balancing problems. In total, this study tests four multi-objective algorithms: MSA, OMABC, the proposed NSGA-II and the proposed IMABC. All these algorithms utilize the same encoding and decoding procedure in "Encoding and decoding" section and the same neighbor structure in "Neighbor structure" section, and the detailed procedure of these algorithms are available upon request.

Following Janardhanan et al. (2019), Nilakantan et al. (2017a) and many others, this study utilizes the computation time of $nt \cdot nt \cdot \tau$ milliseconds as the termination criterion, where all the algorithms terminate when the elapsed CPU time reaches the given computation time. Here, the value of τ is set to 20, 40 and 60 to observe the algorithms' performance under different computation times. All the algorithms are implemented utilizing C++ programming language and running on a set of virtual computers of a tower type of server. This server is equipped with two Intel Xeon E5-2680 v2 processors at 2.8 GHz and 64 GB RAM memory and each virtual computer has one processor and 4 GB RAM memory.

Evaluation indicators

To evaluate the achieved Pareto archives, this study utilizes two quantitative indicators and one graphical indicator. The two quantitative indicators are hyper volume ratio (HVR) and Unary Epsilon Indicator (I_{ε}^{1}). The HVR and I_{ε}^{1} are selected as they are Pareto compliant quantitative indicators; the other indicators, such as the convergence of the Pareto-optimal solution (CP) and the spread metric (SP), are not Pareto compliant and might provide misleading information (Ciavotta et al. 2013; Nilakantan et al. 2017a). The HVR indicator is ratio of the hyper volume of one Pareto archive (S) and that of the true Pareto archive (P), which is calculated with $HVR = \frac{volume\left(\bigcup_{i=1}^{size(S)} x_i\right)}{volume\left(\bigcup_{j=1}^{size(P)} x_j\right)}$. The I_{ε}^{1} indicator measures the minimum distance between one Pareto archive (S) and the true Pareto archive (P), which is calculated with $I_{\varepsilon}^{1} = I_{\varepsilon}(S,P) = max_{x^2} min_{x^1} min_j \frac{f_j(x^1)}{f_j(x^2)}$. For these two indicators, a large value of HVR or a small value of I_{ε}^{1} indicates a superior Pareto archive and the Pareto archive S is close to the true Pareto archive P when the value of HVR and the value of I_{ε}^{1} are closed to 1.0. To calculate the HVR and I_{ε}^{1}, the true or near-true Pareto archive is needed. This study utilizes the Pareto archive by running all the implemented

Table 4 Pareto solutions by developed mathematical model and proposed algorithm

Developed model					Proposed IMABC			
Num.	f_1	f_2	w_1	CPU(s)	Num.	f_1	f_2	CPU(s)
1	170	30.78	0.73–1.00	777.70	1	170	30.78	4.84
2	188	28.19	0.61–0.72	1154.07	2	182	29.5	4.84
3	244	23.49	0.42–0.60	930.40	3	183	29.47	4.84
4	288	21.78	0.38–0.41	678.85	4	188	28.19	4.84
5	367	19.19	0.26–0.37	663.69	5	199	27.79	4.84
6	459	17.48	0.01–0.25	450.85	6	205	26.91	4.84
					7	219	26.48	4.84
					8	225	25.2	4.84
					9	243	24.77	4.84
					10	244	23.49	4.84
					11	288	21.78	4.84
					12	336	20.9	4.84
					13	346	20.47	4.84
					14	367	19.19	4.84
					15	459	17.48	4.84

algorithms in ten times with an elapsed computation time of $nt \cdot nt \cdot 100$ milliseconds as the near-true Pareto archive. Detailed description of the two indicates please refer Nilakantan et al. (2017a) and Ciavotta et al. (2013).

The graphical indicator, Empirical Attainment Function, is utilized to observe the spatial behavior of the tested algorithms in the objective space. Here,$\mathbb{R}^d$ is the d-objective solution space, x_1 is an arbitrary in the solution space $\mathbb{R}^d$ ($x_1 \in \mathbb{R}^d$), Γ is the Pareto archive by algorithm α in one run ($\Gamma = \{x_l \in \mathbb{R}^d, l = 1, 2, \dots, |P_\alpha|\}$). For clarity, the Attainment Function is first described in Eq. (16), which describe the probability that an arbitrary point x_1 is weakly dominated by at least one solution in the Pareto archive by algorithm α in one run. As the probability is unknown, the Empirical Attainment Function (EAF) is proposed with Eqs. (17) and (18), where the probability is empirically approximated by running the algorithm for many times. In Eq. (17), Γ_h is the Pareto archive in h'th runs, nh is the number of repetitions. To evaluate several algorithms, the difference between two EAFs, referred to as $DEAF$, is achieved utilizing Eq. (19). The value of $DEAF$ can be positive when algorithm α has the larger probability or negative when algorithm β has the larger probability. More comprehensive description of this graphical indicator refers to Nilakantan et al. (2017a) and Ciavotta et al. (2013).

$$AF_\alpha(\omega) = P\left(\exists x_1 \in \Gamma : x_1 \trianglelefteq \omega\right) \tag{16}$$

$$EAF_\alpha(\omega) = \frac{1}{nh}\sum_{h=1}^{nh} I\left(\Gamma_h \trianglelefteq \omega\right) \tag{17}$$

$$I\left(\Gamma_h \trianglelefteq \omega\right) = \begin{cases} 1 & if\ \Gamma_h \trianglelefteq \omega \\ 0 & othewise \end{cases} \tag{18}$$

$$DEAF_{\alpha,\beta}(\omega) = \frac{1}{nh}\sum_{h=1}^{nh}\left[I\left(\Gamma_h^\alpha \trianglelefteq \omega\right) - I\left(\Gamma_h^\beta \trianglelefteq \omega\right)\right] \tag{19}$$

The values of the parameters have important effects on the algorithm's performance. Hence, this study calibrates the parameters with the full factorial design of experiments following Nilakantan et al. (2017a) and many others. Firstly, ten different instances are solved for ten times by all the combinations of the parameter values. After completing the experiments, the well-known multi-factor analysis of variance (ANOVA) technique is conducted, where the parameters are regarded as the controlled variable and the average value of $1 - HVR$ in one run is regarded as the response variable. Here, this study replaces the HVR with $1 - HVR$ for better analysis and the parameter value with the smaller value of $1 - HVR$ is preferred. Detailed ANOVA results and the final parameter values are not exhibited for space reasons, but they are available upon request.

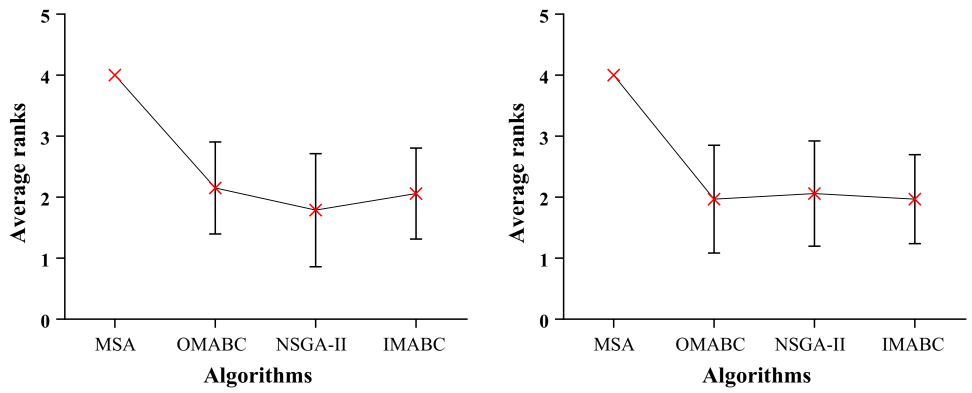
Model evaluation

To evaluate the developed mathematical model, this section solves one instance with 11 tasks and tests a total number of 100 combinations of the w_1 and w_2, where $w_2 = 1 - w_1$. The model is solved with CPLEX solver of General Algebraic Modeling System 23.0 and the model terminates when the optimal solution is found and verified. In preliminary experiments, other instances with more tasks have also been tested, but the model cannot achieve the optimal solution within an

Table 5 Detailed results by tested algorithms under $\tau = 60$

Instance	ns	*HVR*				I_{ε}^{1}			
		MSA	OMABC	NSGA-II	IMABC	MSA	OMABC	NSGA-II	IMABC
P11	4	0.998	1.000	1.000	1.000	1.018	1.006	1.003	1.005
P25	3	0.993	0.997	0.998	0.998	1.019	1.012	1.012	1.013
P25	4	0.990	0.999	0.999	0.998	1.023	1.007	1.005	1.007
P25	6	0.995	0.998	0.999	0.998	1.017	1.012	1.009	1.008
P25	9	0.979	0.994	0.995	0.995	1.057	1.027	1.023	1.027
P35	4	0.989	0.996	0.997	0.998	1.035	1.019	1.014	1.018
P35	5	0.964	0.991	0.989	0.989	1.076	1.027	1.032	1.036
P35	7	0.963	0.999	0.999	0.998	1.159	1.062	1.065	1.046
P35	12	0.934	0.980	0.973	0.978	1.127	1.052	1.075	1.053
P53	5	0.993	0.997	0.997	0.997	1.021	1.011	1.010	1.010
P53	7	0.976	0.995	0.995	0.995	1.040	1.021	1.022	1.021
P53	10	0.941	0.990	0.990	0.991	1.097	1.046	1.045	1.044
P53	14	0.913	0.988	0.988	0.987	1.105	1.028	1.029	1.032
P70	7	0.943	0.984	0.984	0.985	1.074	1.037	1.040	1.036
P70	10	0.950	0.989	0.987	0.988	1.081	1.029	1.034	1.032
P70	14	0.946	0.988	0.990	0.989	1.069	1.028	1.027	1.026
P70	19	0.898	0.977	0.978	0.980	1.239	1.061	1.061	1.054
P89	8	0.954	0.994	0.994	0.995	1.090	1.020	1.018	1.017
P89	12	0.951	0.976	0.970	0.976	1.700	1.379	1.452	1.353
P89	16	0.908	0.987	0.988	0.988	1.120	1.032	1.034	1.034
P89	21	0.899	0.988	0.994	0.994	1.156	1.021	1.015	1.018
P111	9	0.898	0.980	0.973	0.979	1.319	1.088	1.102	1.079
P111	13	0.919	0.985	0.983	0.987	1.151	1.039	1.042	1.040
P111	17	0.907	0.992	0.991	0.991	1.141	1.020	1.021	1.024
P111	22	0.900	0.990	0.988	0.988	1.154	1.025	1.028	1.024
P148	10	0.926	0.991	0.989	0.989	1.097	1.034	1.034	1.036
P148	14	0.897	0.995	0.994	0.994	1.183	1.018	1.019	1.019
P148	21	0.906	0.988	0.988	0.984	1.159	1.026	1.028	1.040
P148	29	0.860	0.986	0.992	0.988	1.207	1.024	1.020	1.024
P297	19	0.886	0.995	0.994	0.997	1.202	1.016	1.014	1.010
P297	29	0.882	0.988	0.992	0.993	1.189	1.027	1.024	1.022
P297	38	0.789	0.966	0.967	0.970	1.253	1.102	1.113	1.119
P297	50	0.778	0.960	0.988	0.979	1.342	1.054	1.025	1.039
Avg	0.928	0.989	0.989	**0.990**	1.143	1.043	1.045	**1.041**	

Best in bold



(a) Ranks of algorithm in terms of $1 - HVR$ **(b)** Ranks of algorithm in terms of I_{ε}^{1}

Fig. 4 Means plot of the average ranks and 95% confidence intervals of algorithms under $\tau = 20$

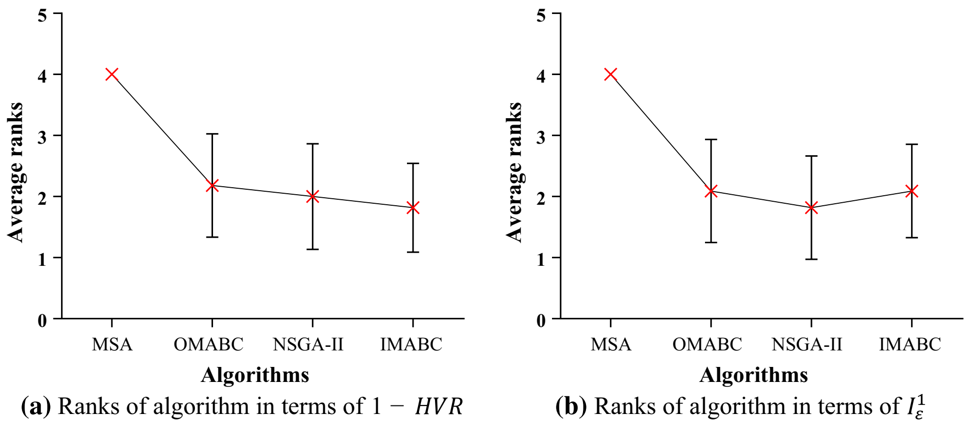


Fig. 5 Means plot of the average ranks and 95% confidence intervals of algorithms under $\tau = 40$

(a) Ranks of algorithm in terms of $1 - HVR$ **(b)** Ranks of algorithm in terms of I_{ε}^{1}

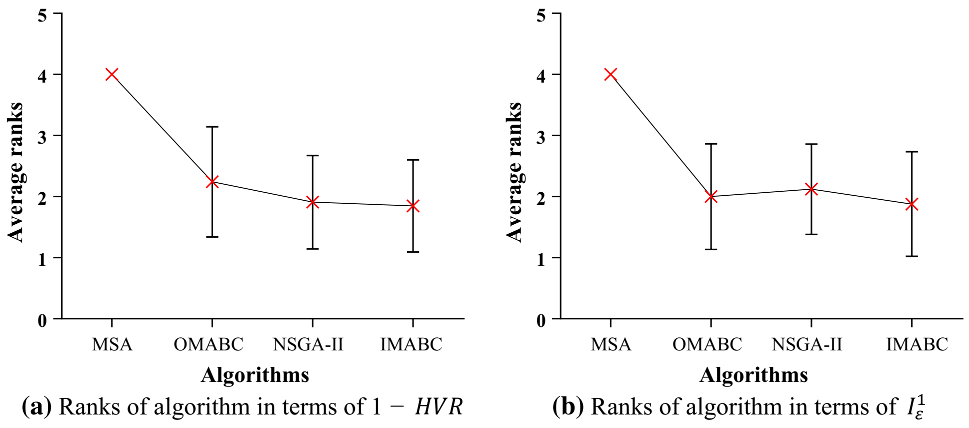


Fig. 6 Means plot of the average ranks and 95% confidence intervals of algorithms under $\tau = 60$

(a) Ranks of algorithm in terms of $1 - HVR$ **(b)** Ranks of algorithm in terms of I_{ε}^{1}

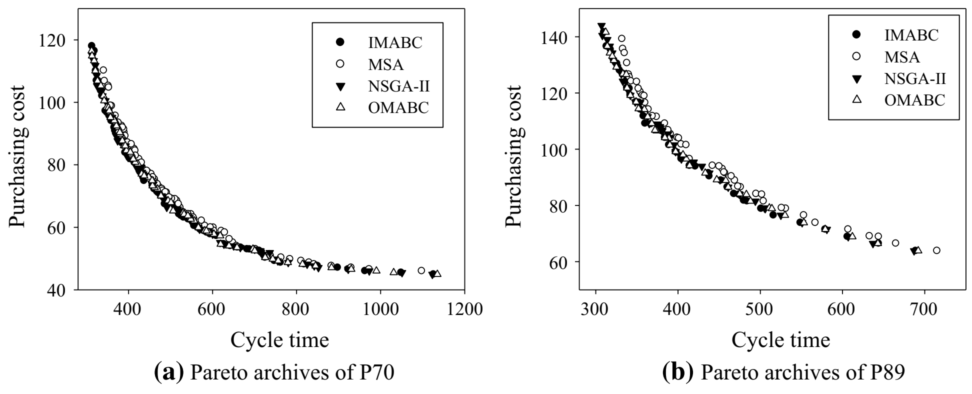


Fig. 7 Pareto archives of P70 and P89 by the algorithms within ten runs

(a) Pareto archives of P70 **(b)** Pareto archives of P89

acceptable time. Hence, this model only solves the smallest-size instance here.

Table 4 presents the achieved Pareto solutions by the model and proposed IMABC algorithm in ten repetitions under $\tau = 60$. It is observed that the developed mathematical model is capable of achieving six Pareto solutions by testing 100 combinations of the w_1 and w_2 whereas the proposed IMABC is capable of achieving a total number of 15 Pareto solutions. All Pareto solutions achieved by the model belong to the Pareto archive by the proposed IMABC, demonstrating that the proposed IMABC is more efficient and effective in solving multi-objective optimization.

To summarize, three findings could be achieved from this comparative study as follows. (1) The utilization of multi-objective optimization is rational. The sole optimization of cycle time cannot optimize the total purchasing cost

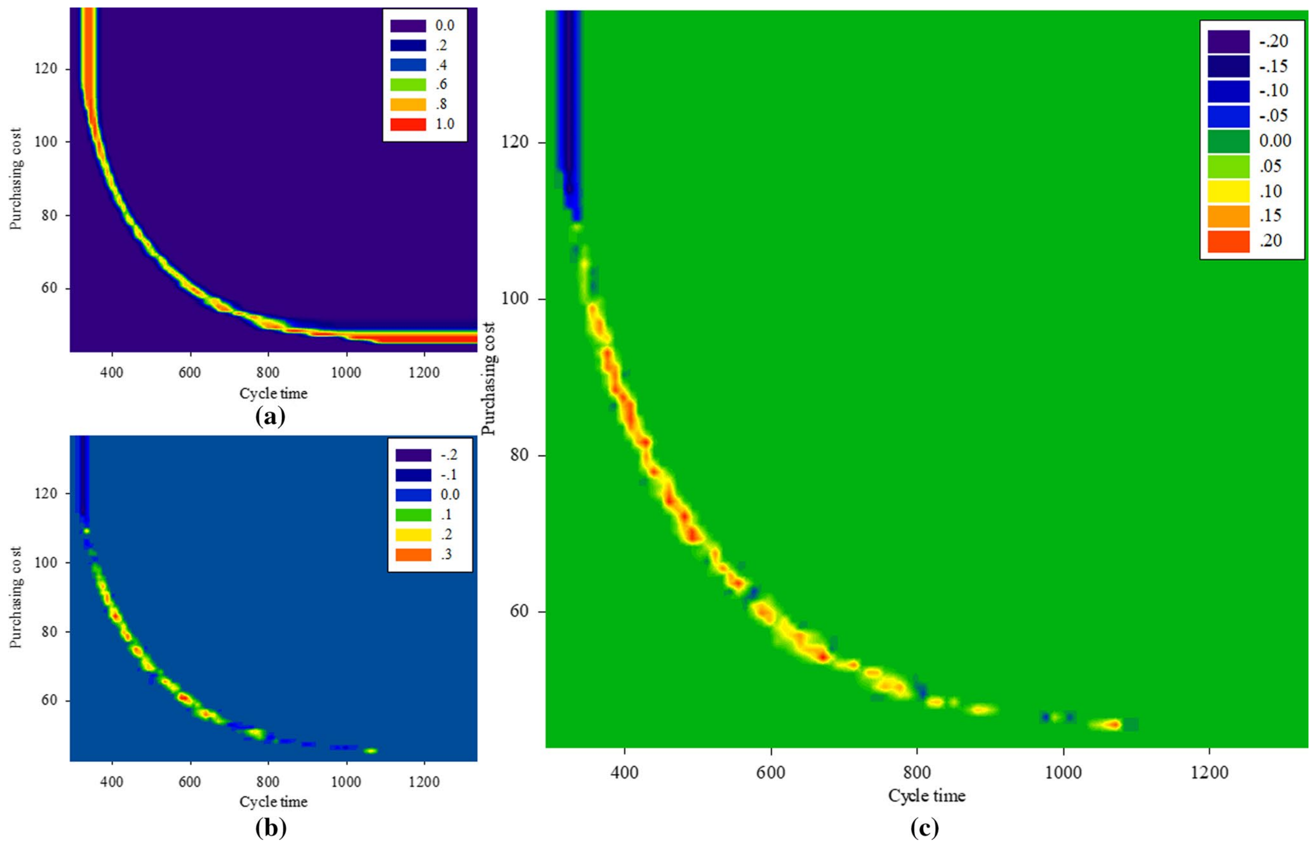


Fig. 8 Differences between Empirical Attainment Functions between IMABC and MSA (**a**), IMABC and OMABC (**b**), and IMABC and NSGA-II (**c**)

effectively; the sole optimization of the purchasing cost cannot optimize the cycle time effectively. (2) The developed mathematical model is capable of achieving the optimal solution with the weighted objective and it is capable of achieving Pareto solutions by testing many combinations of the w_1 and w_2. (3) The proposed multi-objective algorithm is capable of achieving more Pareto solutions with short computation time, and the multi-objective algorithm is more effective and efficient than the formulated model.

Comparative study

This section evaluates all the implemented algorithms in terms of the *HVR* and I_{ε}^{1}. Table 5 illustrates the detailed results obtained by the four algorithms under $\tau = 60$, where each cell reports the average results in ten repetitions. It can be seen from this table that IMABC is the best performer in terms of the *HVR* and I_{ε}^{1}, NSGA-II and OMABC produces similar performance and MSA is the worst performer in terms of the two evaluation indicators. It can be noted that MSA outperforms NSGA-II in studies by Li et al. (2016b) and Nilakantan et al. (2017a) whereas NSGA-II outperforms the MSA in this study. The main reason lies behind that this study utilizes different encoding scheme and different decoding procedure, and the computational study demonstrates that the population algorithms are more suitable for the considered problem.

To evaluate the algorithms statistically, this study conducts the non-parametric Friedman rank-based analysis as the normality is slightly violated in the preliminary ANOVA test. Figures 4, 5 and 6 depicts the ranks of the algorithm in terms of $1 - HVR$ and I_{ε}^{1} under $\tau = 20$, $\tau = 40$ and $\tau = 60$. From Fig. 6, it is clear that OMABC, NSGA-II and IMABC produces similar performance, but they produce statistically superior performances than MSA in terms of $1 - HVR$ and I_{ε}^{1}. The IMABC obtains the peak performance and slightly better performance than OMABC and NSGA-II under $\tau = 60$ in accordance with the results in Table 5.

To have a better observation of the achieved Pareto archives by the algorithms, Fig. 7 presents the Pareto archives by algorithms for ten times in solving the P70 with 14 stations and P89 with 16 stations under $\tau = 60$. From Fig. 7a and b, it is observed that IMABC, MSA and OMABC achieve ambiguous performance, but they obtain clear better Pareto archive than the MSA.

To further observe the spatial performance, Fig. 8 depicts the *DEAF* between IMABC and MSA (a), IMABC and OMABC (b) and IMABC and NSGA-II (c) in solving the P70 with 14 stations under $\tau = 60$. It is to be noted that all the algorithms are running for 100 times to achieve these figures. Regarding Fig. 8a, if the value in one objective area is larger than 0.0, IMABC performs better than MSA in this objective area; if the value in one objective area is less than 0.0, MSA performs better than IMABC in this objective area. Clearly, IMABC achieve better performance and IMABC outperforms or achieve the same performance in all the objective space. And it is observed in Fig. 8b that IMABC outperforms OMABC in the central part of the objective space whereas OMABC performs better at the extreme points of the Pareto archive. Similarly, IMABC outperforms the NSGA-II in the central part of the objective space whereas NSGA-II performs better at the extreme points of the Pareto archive in Fig. 8c. The graphic tool demonstrates the superiority of the developed IMABC over the MSA and the OMABC, NSGA-II and IMABC obtain ambiguous performances, where IMABC performs better at the center points of the Pareto archive.

Extension of the model and algorithms for type-I RALBP

As type-I RALBP happens when there is no space limit and there is no limit of the number of robots, this section also extends the developed model and algorithms to the type-I RALBP. For clarification, the new and modified notations are first introduced as follows.

Indices	
k	Station index, $k \in \{1, 2, \ldots, UB_{ns}\}$, where UB_{ns} is the upper bound of the number of stations.
K	Set of stations, $K = \{1, 2, \ldots, UB_{ns}\}$.
Parameters	
CT	The given cycle time
Decision variables	
v_k	1, if there is a least on task allocated to station k; 0, otherwise.

On the basis of the model in "Mathematical model" section, the model for type-I RALBP is formulated with expression (20)–(33). Here, the objective 1 in expression (20) minimizes the number of stations and the objective 2 in expression (21) minimizes the total purchasing cost. Constraint (22) deals with the assignment of tasks and constraint (23) indicates that one station is allocated with one robot if this station is opened. The explanations of the expression (24) to expression (31) are the same to that in "Mathematical model" section. Constraint (32) indicates that one station is opened if there is at least one task allocated to this station. It can be noted that this new formulated model is also a MILP model and it can solve the small-size instances optimally when optimizing either objective or combining the two objectives into one weighted objective.

$$\mathrm{Min}\, f_1 = \sum_{k \in K} v_k \tag{20}$$

$$\mathrm{Min}\, f_2 = \sum_{r \in R} \sum_{k \in K} c_r w_{rk} \tag{21}$$

$$\sum_{r \in R} \sum_{k \in K} \sum_{p=1}^{Nm_k} \mathrm{x}_{irkp} = 1 \quad \forall \mathrm{i} \in \mathrm{I} \tag{22}$$

$$\sum_{r \in R} w_{rk} = v_k \forall k \in K \tag{23}$$

$$\sum_{i \in I} \mathrm{x_{irkp}} \leq \psi \cdot w_{rk} \; \forall r \in R, k \in K, \quad \mathrm{p} = 1, \ldots, Nm_k \tag{24}$$

$$\sum_{i \in I} \mathrm{x}_{irkp} \leq 1 \; \forall \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \quad \mathrm{p} = 1, \ldots, Nm_k \tag{25}$$

$$\sum_{i \in I} \mathrm{x_{irk,p+1}} \leq \sum_{i \in I} \mathrm{x_{irkp}} \; \forall \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \quad \mathrm{p} = 1, \ldots, Nm_k - 1 \tag{26}$$

$$\sum_{r \in R} \sum_{k \in K} \sum_{p=1}^{Nm_k} (N^{max} \cdot (k-1) + p) \cdot \mathrm{x_{irkp}} \leq \sum_{r \in R} \sum_{k \in K} \sum_{p=1}^{Nm_k} (N^{max} \cdot (k-1) + p) \cdot \mathrm{x_{jrkp}} \quad \forall (i, \mathrm{j}) \in \wp \tag{27}$$

$$\sum_{i \in I} \sum_{r \in R} \sum_{p=1}^{Nm_k} t_{ir} \cdot \mathrm{x_{irkp}} + \sum_{i \in I} \sum_{\mathrm{j} \in \mathrm{I} \wedge (i \neq j)} \sum_{r \in R} s_{ijr} \cdot \mathrm{z_{ijrk}} \leq CT \quad \forall k \in K \tag{28}$$

$$\mathrm{x_{irkp}} + \mathrm{x_{jrk,p+1}} \leq 1 + z_{ijrk} \; \forall \mathrm{i,j} \in \mathrm{I} \wedge (\mathrm{i} \neq \mathrm{j}) \wedge (\mathrm{j} \notin PT_i), \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \; \mathrm{p} = 1, \ldots, Nm_k - 1 \tag{29}$$

$$\mathrm{x_{irkp}} - \sum_{\forall \mathrm{j} \in \mathrm{I} | (i \neq j) \wedge (\mathrm{j} \notin PT_i)} \mathrm{x_{jrk,p+1}} \leq y_{irk} \quad \forall \mathrm{i} \in \mathrm{I}, \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K}, \quad \mathrm{p} = 1, \ldots, Nm_k - 1 \tag{30}$$

$$y_{irk} + \mathrm{x_{jrk1}} \leq 1 + z_{ijrk} \; \forall \mathrm{i,j} \in \mathrm{I} \wedge (\mathrm{i} \neq \mathrm{j}) \wedge (\mathrm{i} \notin PT_j), \mathrm{r} \in \mathrm{R}, \mathrm{k} \in \mathrm{K} \tag{31}$$

$$v_k \geq \max_{i \in I} \left(\sum_{r \in R} \sum_{p=1}^{Nm_k} x_{irkp} \right) \tag{32}$$

$$x_{irkp}, y_{irk}, z_{ijrk}, v_k, w_{rk} \in \{0, 1\} \tag{33}$$

As for the algorithms to solve type-I RALBP, the similar encoding scheme with task permutation vector and robot selection vector can be utilized here. Notice that, for the robot selection vector, a number of UB_{ns} elements is utilized and code r in position k donates that robot r is allocated to station k. The decoding procedure is much easier as the cycle time is given and it needs only one time of decoding procedure to achieve the objective functions. Afterwards, the developed multi-objective algorithms can be applied directly to obtain a set of Pareto solutions. Or it is also applicable to solve one weighted objective with the one-objective algorithms in Janardhanan et al. (2019) and Li et al. (2019b) directly with the new encoding and decoding procedure.

Conclusions and future research

Modern industry utilizes the robots to replace human workers for the purpose of achieving higher productivity and lower cost. The purchasing cost of robots is very important when designing the robotic assembly line. This paper provides the first study to tackle the multi-objective cost-oriented robotic assembly line balancing problem with setup times to minimize the cycle time and total purchasing cost simultaneously. Firstly, a mixed-integer linear programming model is formulated, and the developed model is capable of solving the small-size problems optimally utilizing the CPLEX solver when combining the two objectives into one weighted objective. As two objectives conflict in many occasions, this study develops the elitist non-dominated sorting genetic algorithm (NSGA-II) and improved multi-objective artificial bee colony (IMABC) algorithm to achieve a set of Pareto solutions for line manager to select. The proposed IMABC utilizes crossover operator in the employed bee phase to emphasize exploration and utilize new scout phase, which selects one solution in the permanent Pareto archive to replace the abandoned solution, to emphasize exploitation. This study also conducts a comprehensive study to evaluate the performance of the proposed multi-objective algorithms, where they are compared with the multi-objective simulated annealing algorithms and the original multi-objective artificial bee colony algorithms. Comparative study on a set of generated instances verifies that the proposed algorithms achieve competing performance in comparison with other algorithms.

The developed model and algorithms can provide a set of high-quality Pareto solutions for the production managers to select. The proposed algorithms might be integrated into the decision support system to assist the manager to design effective and high-quality robotic assembly lines.

In future research, the cost model can be extended to U-shaped robotic assembly line, parallel robotic assembly line, two-sided robotic assembly line and others. The developed multi-objective algorithms can be also employed to solve other assembly line balancing problems. It might be a good choice to utilize the sliding frame technique presented in Cohen and Dar-El (2010) to dissect the original problem to small controllable-size sub-problems when solving the large-size instances. Another interesting topic is taking into account the maintenance of the robots when selecting the robots. For instance, if most stations have two types of robots, considering one of them for the next station has a lower maintenance cost than deploying a new type of robot. In real application, the robots are utilized to partially replace the human worker or assist the human workers in many circumstances. Hence, it is also suggested to study the cooperation between the human workers and robots and surely the developed cost model could also be applied when selecting the robots on this type of line.

Acknowledgements This project is partially supported by National Natural Science Foundation of China under Grant 61803287 and China Postdoctoral Science Foundation under Grant 2018M642928.

References

Aghajani, M., Ghodsi, R., & Javadi, B. (2014). Balancing of robotic mixed-model two-sided assembly line with robot setup times. *International Journal of Advanced Manufacturing Technology, 74*(5–8), 1005–1016. https://doi.org/10.1007/s00170-014-5945-x.

Akpinar, Ş., & Baykasoğlu, A. (2014a). Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model. *Journal of Manufacturing Systems, 33*(1), 177–187. https://doi.org/10.1016/j.jmsy.2013.11.004.

Akpinar, Ş., & Baykasoğlu, A. (2014b). Modeling and solving mixed-model assembly line balancing problem with setups. Part II: A multiple colony hybrid bees algorithm. *Journal of Manufacturing Systems, 33*(4), 445–461.

Akpınar, S., Mirac Bayhan, G., & Baykasoglu, A. (2013). Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Applied Soft Computing, 13*(1), 574–589. https://doi.org/10.1016/j.asoc.2012.07.024.

Amen, M. (2000). Heuristic methods for cost-oriented assembly line balancing: A survey. *International Journal of Production Economics, 68*(1), 1–14. https://doi.org/10.1016/S0925-5273(99)00095-X.

Amen, M. (2001). Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. *International Journal of Production Economics, 69*(3), 255–264. https://doi.org/10.1016/S0925-5273(99)00096-1.

Amen, M. (2006). Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. *European*

Journal of Operational Research, 168(3), 747–770. https://doi.org/10.1016/j.ejor.2004.07.026.

Andrés, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research, 187*(3), 1212–1223. https://doi.org/10.1016/j.ejor.2006.07.044.

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics, 142*(2), 259–277.

Borba, L., Ritt, M., & Miralles, C. (2018). Exact and heuristic methods for solving the robotic assembly line balancing problem. *European Journal of Operational Research, 270*(1), 146–156. https://doi.org/10.1016/j.ejor.2018.03.011.

Ciavotta, M., Minella, G., & Ruiz, R. (2013). Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. *European Journal of Operational Research, 227*(2), 301–313. https://doi.org/10.1016/j.ejor.2012.12.031.

Çil, Z. A., Mete, S., & Ağpak, K. (2017a). Analysis of the type II robotic mixed-model assembly line balancing problem. *Engineering Optimization, 49*(6), 990–1009. https://doi.org/10.1080/0305215X.2016.1230208.

Çil, Z. A., Mete, S., Özceylan, E., & Ağpak, K. (2017b). A beam search approach for solving type II robotic parallel assembly line balancing problem. *Applied Soft Computing, 61,* 129–138. https://doi.org/10.1016/j.asoc.2017.07.062.

Cohen, Y., & Dar-El, E. (2010). The sliding frame-extending the concept to various assembly line balancing problems. *International Journal of Manufacturing Technology and Management, 20*(1/2/3/4), 4–24.

Daoud, S., Chehade, H., Yalaoui, F., & Amodeo, L. (2014). Solving a robotic assembly line balancing problem using efficient hybrid methods. *Journal of Heuristics, 20*(3), 235–259. https://doi.org/10.1007/s10732-014-9239-0.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Delice, Y. (2019). A genetic algorithm approach for balancing two-sided assembly lines with setups. *Assembly Automation, 39*(5), 827–839. https://doi.org/10.1108/AA-11-2018-0192.

Foroughi, A., & Gökçen, H. (2019). A multiple rule-based genetic algorithm for cost-oriented stochastic assembly line balancing problem. *Assembly Automation, 39*(1), 124–139. https://doi.org/10.1108/aa-03-2018-050.

Gao, J., Sun, L., Wang, L., & Gen, M. (2009). An efficient approach for type II robotic assembly line balancing problems. *Computers & Industrial Engineering, 56*(3), 1065–1080. https://doi.org/10.1016/j.cie.2008.09.027.

Hamta, N., Fatemi Ghomi, S. M. T., Jolai, F., & Akbarpour Shirazi, M. (2013). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics, 141*(1), 99–111. https://doi.org/10.1016/j.ijpe.2012.03.013.

Janardhanan, M. N., Li, Z., Bocewicz, G., Banaszak, Z., & Nielsen, P. (2019). Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times. *Applied Mathematical Modelling, 65,* 256–270. https://doi.org/10.1016/j.apm.2018.08.016.

Kim, H., & Park, S. (1995). A strong cutting plane algorithm for the robotic assembly line balancing problem. *International Journal of Production Research, 33*(8), 2311–2323. https://doi.org/10.1080/00207549508904817.

Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*(3), 811–825. https://doi.org/10.1016/j.ejor.2004.07.030.

Li, Z., Dey, N., Ashour, A. S., & Tang, Q. (2018a). Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem. *Neural Computing and Applications, 30*(9), 2685–2696. https://doi.org/10.1007/s00521-017-2855-5.

Li, Z., Janardhanan, M. N., Ashour, A. S., & Dey, N. (2019a). Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-018-3957-4.

Li, Z., Janardhanan, M. N., Nielsen, P., & Tang, Q. (2018b). Mathematical models and simulated annealing algorithms for the robotic assembly line balancing problem. *Assembly Automation, 38*(4), 420–436. https://doi.org/10.1108/Aa-09-2017-115.

Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (2018c). Mathematical model and metaheuristics for simultaneous balancing and sequencing of a robotic mixed-model assembly line. *Engineering Optimization, 50*(5), 877–893. https://doi.org/10.1080/0305215x.2017.1351963.

Li, Z., Janardhanan, M. N., Tang, Q., & Ponnambalam, S. G. (2019b). Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. *Swarm and Evolutionary Computation, 50,* 100567. https://doi.org/10.1016/j.swevo.2019.100567.

Li, Z., Nilakantan, J. M., Tang, Q., & Nielsen, P. (2016a). Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Advances in Mechanical Engineering, 8*(9), 1–14. https://doi.org/10.1177/1687814016667907.

Li, Z., Tang, Q., & Zhang, L. (2016b). Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm. *Journal of Cleaner Production, 135,* 508–522. https://doi.org/10.1016/j.jclepro.2016.06.131.

Nilakantan, J. M., Huang, G. Q., & Ponnambalam, S. (2015a). An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *Journal of Cleaner Production, 90,* 311–325.

Nilakantan, M., Li, Z., Tang, Q., & Nielsen, P. (2017a). Multi-objective co-operative co-evolutionary algorithm for minimizing carbon footprint and maximizing line efficiency in robotic assembly line systems. *Journal of Cleaner Production, 156,* 124–136. https://doi.org/10.1016/j.jclepro.2017.04.032.

Nilakantan, M., Nielsen, I., Ponnambalam, S. G., & Venkataramanaiah, S. (2017b). Differential evolution algorithm for solving RALB problem using cost- and time-based models. *The International Journal of Advanced Manufacturing Technology, 89*(1), 311–332. https://doi.org/10.1007/s00170-016-9086-2.

Nilakantan, J. M., & Ponnambalam, S. (2016). Robotic U-shaped assembly line balancing using particle swarm optimization. *Engineering Optimization, 48*(2), 231–252.

Nilakantan, M., Ponnambalam, S., & Jawahar, N. (2016). Design of energy efficient RAL system using evolutionary algorithms. *Engineering Computations, 33*(2), 580–602.

Nilakantan, M., Ponnambalam, S. G., Jawahar, N., & Kanagaraj, G. (2015b). Bio-inspired search algorithms to solve robotic assembly line balancing problems. *Neural Computing and Applications, 26*(6), 1379–1393. https://doi.org/10.1007/s00521-014-1811-x.

Özcan, U. (2019). Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. *International Journal of Production Economics, 213,* 81–96. https://doi.org/10.1016/j.ijpe.2019.02.023.

Özcan, U., & Toklu, B. (2010). Balancing two-sided assembly lines with sequence-dependent setup times. *International Journal of Production Research, 48*(18), 5363–5383. https://doi.org/10.1080/00207540903140750.

Padrón, M., Irizarry, M. A., Resto, P., & Mejía, H. P. (2009). A methodology for cost-oriented assembly line balancing problems. *Journal of Manufacturing Technology Management, 20*(8), 1147–1165. https://doi.org/10.1108/17410380910997254.

Pereira, J., Ritt, M., & Vásquez, Ó. C. (2018). A memetic algorithm for the cost-oriented robotic assembly line balancing problem. *Computers & Operations Research, 99,* 249–261. https://doi.org/10.1016/j.cor.2018.07.001.

Rabbani, M., Mousavi, Z., & Farrokhi-Asl, H. (2016). Multi-objective metaheuristics for solving a type II robotic mixed-model assembly line balancing problem. *Journal of Industrial and Production Engineering, 33*(7), 472–484. https://doi.org/10.1080/21681015.2015.1126656.

Roshani, A., Fattahi, P., Roshani, A., Salehi, M., & Roshani, A. (2012). Cost-oriented two-sided assembly line balancing problem: A simulated annealing approach. *International Journal of Computer Integrated Manufacturing, 25*(8), 689–715. https://doi.org/10.1080/0951192X.2012.664786.

Rubinovitz, J., & Bukchin, J. (1991). *Design and balancing of robotic assembly lines*. Dearborn: Society of Manufacturing Engineers.

Rubinovitz, J., Bukchin, J., & Lenz, E. (1993). RALB—a heuristic algorithm for design and balancing of robotic assembly lines. *CIRP Annals, 42*(1), 497–500. https://doi.org/10.1016/s0007-8506(07)62494-9.

Şahin, M., & Kellegöz, T. (2017). Increasing production rate in U-type assembly lines with sequence-dependent set-up times. *Engineering Optimization, 49*(8), 1401–1419. https://doi.org/10.1080/0305215X.2016.1256394.

Saif, U., Guan, Z., Liu, W., Wang, B., & Zhang, C. (2014). Multi-objective artificial bee colony algorithm for simultaneous sequencing and balancing of mixed model assembly line. *The International Journal of Advanced Manufacturing Technology, 75*(9–12), 1809–1827.

Saif, U., Guan, Z., Zhang, L., Zhang, F., Wang, B., & Mirza, J. (2019). Multi-objective artificial bee colony algorithm for order oriented simultaneous sequencing and balancing of multi-mixed model assembly line. *Journal of Intelligent Manufacturing, 30*(3), 1195–1220. https://doi.org/10.1007/s10845-017-1316-4.

Salehi, M., Maleki, H. R., & Niroomand, S. (2019). Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-019-04293-8.

Scholl, A., Boysen, N., & Fliedner, M. (2008). The sequence-dependent assembly line balancing problem. *OR Spectrum, 30*(3), 579–609. https://doi.org/10.1007/s00291-006-0070-3.

Scholl, A., Boysen, N., & Fliedner, M. (2013). The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum, 35*(1), 291–320. https://doi.org/10.1007/s00291-011-0265-0.

Seyed-Alagheband, S. A., Ghomi, S. M. T. F., & Zandieh, M. (2011). A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. *International Journal of Production Research, 49*(3), 805–825. https://doi.org/10.1080/00207540903471486.

Yolmeh, A., & Kianfar, F. (2012). An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Computers & Industrial Engineering, 62*(4), 936–945. https://doi.org/10.1016/j.cie.2011.12.017.

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., & Ardakani, H. D. (2012). Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. *Journal of Manufacturing Systems, 31*(2), 139–151. https://doi.org/10.1016/j.jmsy.2011.10.002.

Zhang, Z., Tang, Q., Li, Z., & Zhang, L. (2018). Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems. *International Journal of Production Research, 57*(17), 5520–5537. https://doi.org/10.1080/00207543.2018.1530479.

Zhang, Z., Tang, Q., & Zhang, L. (2019). Mathematical model and grey wolf optimization for low-carbon and low-noise U-shaped robotic assembly line balancing problem. *Journal of Cleaner Production, 215,* 744–756. https://doi.org/10.1016/j.jclepro.2019.01.030.

Zhou, B., & Wu, Q. (2019). An improved immune clonal selection algorithm for bi-objective robotic assemble line balancing problems considering time and space constraints. *Engineering Computations, 36*(6), 1868–1892. https://doi.org/10.1108/Ec-11-2018-0512.