Neural Computing and Applications (2021) 33:8575–8596
https://doi.org/10.1007/s00521-020-05610-2

ORIGINAL ARTICLE

Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots

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

Received: 8 May 2020 / Accepted: 11 December 2020 / Published online: 3 January 2021

Abstract
Industries are increasingly looking for opportunities at utilizing collaborative robots in assembly lines to perform the tasks independently or assist the human workers due to the advancement of industry 4.0 technologies. Purchasing cost is one of the important factors to be considered by production managers, while designing or redesigning assembly line when collaborative robots are being utilized. Several objectives are to be optimized in an assembly line balancing problem and optimizing line efficiency along with purchasing cost sometimes results in conflicting situation. This paper presents the first study to tackle the cost-oriented assembly line balancing problem with collaborative robots, where several different types of collaborative robots with different purchasing costs are available and selected. A multi-objective mixed-integer programming model is developed to minimize the cycle time and the total collaborative robot purchasing cost. The multi-objective migrating bird optimization algorithm is developed to obtain a set of high-quality Pareto solutions. This algorithm utilizes the fast non-dominated sorting approach to update the population and develops a restart mechanism to select one solution in the permanent Pareto archive to replace the abandoned solution which remains unchanged for several iterations. The computational study validates that the utilization of the multi-objective model is reasonable and developed algorithm produces competing performance in comparison with multi-objective non-dominated sorting genetic algorithm II, multi-objective simulated annealing algorithm and two multi-objective artificial bee colony algorithms.

Keywords Assembly line balancing · Human–robot collaboration · Collaborative robots · Multi-objective optimization · Migrating bird optimization

Abbreviations

ALBP	Assembly line balancing problem
RALBP	Robotic assembly line balancing problem
CRALBP	Assembly line balancing problem with collaborative robots
MILP	Mixed-integer linear programming
Cobot	Collaborative robot
MBO	Migrating bird optimization
MMBO	Multi-objective migrating bird optimization
MSA	Multi-objective simulated annealing algorithm
NSGA-II	Non-dominated sorting genetic algorithm II
OMABC	Original multi-objective artificial bee colony algorithm
IMABC	Improved multi-objective artificial bee colony algorithm
CP	The convergence of the Pareto-optimal solution
SP	The spread metric
HVR	Hyper volume ratio
I_{ε}^{1}	Unary Epsilon indicator

✉ Mukund Nilakantan Janardhanan
mj251@leicester.ac.uk

Zixiang Li
lizixiang@wust.edu.cn

Qiuhua Tang
tangqiuhua@wust.edu.cn

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

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

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

1 Introduction

Assembly line is the vital and final step to produce the products, and assembly lines have been widely applied due to higher productivity [1]. The assembly line consists of a set of stations, and the assembly tasks are divided and completed by the workers on stations in a sequence. Assembly line balancing problem (ALBP) occurs when designing the assembly line, in which the main focus is to the determine the appropriate task assignment to the stations with one or several optimization criteria under precedence constraint and cycle time constraint.

Along with the increased labour cost and the development automation technology, there is a clear trend to replace the human worker with industrial robots [2]. This new type of assembly line is referred as robotic assembly line which helps to achieve high-volume production with lower costs and higher stability. However, the utilization of robotic assembly line has two possible drawbacks: (1) the cost might be very high to buy and operate these robots, e.g. the purchasing cost of high-tech robots to perform the complicated tasks; (2) the robotic assembly line might lack flexibility to meet personalized customer requirements and rapid market changes. To overcome these drawbacks, the collaborative robots, referred to as collaborative robots (cobots), draws increasing attentions from both industrial and scientific communities to achieve human–robot collaboration. The human–robot collaboration combines the advantages of human workers like higher flexibility, higher adaptability, higher creativity and decision-making skills and the advantages of the cobots like strength, endurance, speed and accuracy [3]. The assembly line with cobots could achieve high production efficiency, high quality and high flexibility, and it also benefits the workers' health when utilizing the cobots to perform the stressful and repetitive tasks.

Figure 1 illustrates the layout of the assembly line with cobots. In the traditional assembly line with workers, the human workers are assigned to the stations to operate the tasks. In the robotic assembly line with the traditional industrial robots, robots are allocated to the stations to operate the tasks and it is not allowed to assign one robot and one worker to one station due to safety reasons. In the assembly line with workers and cobots, one station is allocated with one worker, one cobot or a worker and cobot. And the tasks assigned to the station with one worker and one cobot might be operated by the worker, the cobot or the worker and the cobot collaboratively.

There are many types of cobots available produced by different companies, and different types of cobots have different skills and purchasing costs. When designing the assembly line with cobots, the purchasing cost is one of the most important factors and cannot be ignored. As the high-tech cobots might have the higher cost, there is a tradeoff between line efficiency and the budget. Hence, it is important to study the assembly line balancing problem with collaborative robots (CRALBP), to minimize the cycle time and purchasing cost simultaneously.

In the literature, there are many new metaheuristics designed to tackle the global optimization problems. These metaheuristics include random walk grey wolf optimizer [4], harmonized salp chain-built optimization [5], hybrid grey wolf optimizer with mutation operator [6], memory guided sine cosine algorithm [7], memory-based grey wolf optimizer [8], self-adaptive sine cosine algorithm [9], cauchy Grey Wolf Optimiser [10] and many others. Among these recent and popular metaheuristics, migrating bird optimization (MBO) algorithm is designed on the basis of the V flight formation of the migrating birds and also produces competing performance in solving several variants of robotic assembly line balancing problems such as two-sided [2], U-shaped [11] and robotic assembly lines with setup times [12]. However, there are no reported application of the MBO to solve the considered CRALBP.

Although there are some reported studies in CRALBP, as presented in the literature Sect. 2, this study differentiates the reported ones in both problem setting as well as in the methodology utilized. For instance, the main differences between this study and that by Weckenborg et al. [3] are analyzed as follows: (1) Weckenborg et al. [3] consider

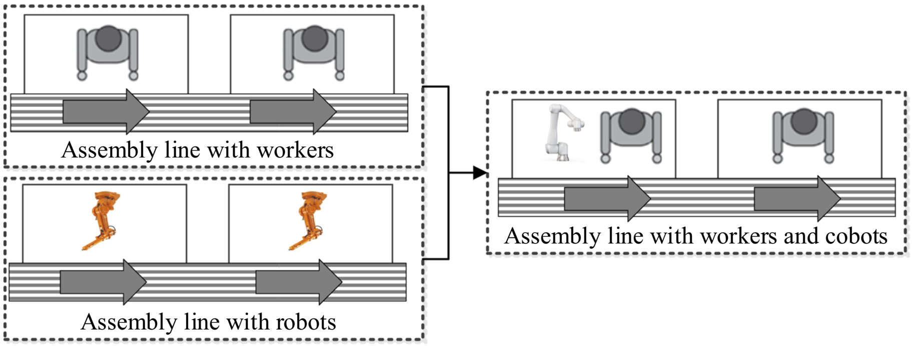
Assembly line with workers
Assembly line with robots
Assembly line with workers and cobots

Fig. 1 Layout of the assembly line with cobots

the situation where only one type of cobot is available, whereas this study takes into account the situation where several types of cobots are available. (2) Weckenborg et al. [3] developed single objective genetic algorithm to minimize the cycle time, whereas this study develops a multi-objective migrating bird optimization algorithm to achieve a set of Pareto solutions for the production line manager to select. Due to the reasons mentioned, this study presents the first study to tackle the CRALBP to minimize the cycle time and purchasing cost simultaneously. The main contributions of this work are listed as follows: (1) This study presents a multi-objective mixed-integer linear programming (MILP) model to minimize the cycle time and purchasing cost. This model can be solved utilizing the CPLEX solver with either objective or weighted objective. To the author's best knowledge, this is the first time to consider the purchasing cost in the CRALBP. (2) This study presents a multi-objective migrating bird optimization (MMBO) algorithm to achieve a set of Pareto solutions for the line manager to select. The proposed MMBO proposes the fast non-dominated sorting method for the population evolution and develops a new restart mechanism to select a solution from the permanent Pareto to replace the individuals which have not been updated in many iterations, with the purpose of helping the algorithm to escape from being trapped into local optima. This is the first time to proposed the MMBO to study the multi-objective CRALBP. (3) This study extends several other multi-objective algorithms to solve the considered problem for the first time, including the multi-objective simulated annealing algorithm, non-dominated sorting genetic algorithm II, two multi-objective artificial bee colony algorithms. This study conducts the comparative study and statistical analysis to evaluate the performances of the implemented methodologies.

The remainder of this study is organized as follows. Section 2 reviews the related literature, and Sect. 3 formulates the studied problem. The developed multi-objective migrating bird optimization algorithm is described in Sect. 4 along with an illustrated example. Section 5 presents the detailed experimental results. Finally, Sect. 6 concludes this study and presents several future research directions.

2 Literature review

ALBPs have drawn increasing attentions from both industrial area [1] and scientific area [13]. As far as the optimization objective is concerned, the ALBPs are mainly divided into three categories: ALBP-I to minimize the number of stations with a given cycle time, ALBP-II to minimize the cycle time with a given number of stations, and ALBP-E to maximize the line efficiency.

This section mainly reviews the recent studies related to the ALBPs, robotic assembly line balancing problems (RALBPs) and the studies related to the CRALBP. The basic version of ALBP is the simple ALBP to optimize the task assignment. Many heuristics, metaheuristics and exact methods have been applied to the ALBP [1]. Among these methods, the high-performing heuristics include the modified Hoffman heuristic [14] and beam search heuristic [15]; the high-performing metaheuristics include the modified Tabu search [16] and beam ant colony optimization [17]; the best-performing exact method is the branch, bound and remember algorithm [18], which produces the state-of-the-art performance. One could refer to the Battaïa, Dolgui [1] for detailed description of the applied methods.

Metaheuristic optimization techniques have been extensively used in solving the global optimization problems due to simplicity, flexibility and derivation-free mechanism. The most popular algorithm is the genetic algorithm, and recently many more powerful algorithms have been developed and they could be mainly divided into three categories: evolution-based algorithms, physics-based algorithm and swarm-based algorithms. The evolution-inspired algorithms include the genetic algorithm, the recent biogeography-based optimizer [19] and others. The physics-based algorithms include the simulated annealing and others. The swarm-based algorithms include the particle swarm optimization, cuckoo search algorithm [20], grey wolf optimizer [21], whale optimization algorithm [22] and many others. There are also some improved versions of these algorithms and they include random walk grey wolf optimizer [4], harmonized salp chain-built optimization [5], hybrid grey wolf optimizer with mutation operator [6], memory guided sine cosine algorithm [7], memory-based grey wolf optimizer [8], self-adaptive sine cosine algorithm [9], cauchy grey wolf optimiser [10] and many others. Among these recent and popular metaheuristics, migrating bird optimization (MBO) algorithm is designed on the basis of the V flight formation of the migrating birds and also produces competing performance in solving assembly line balancing problems [2, 11, 12]. However, there is no reported application of the MBO to solve the considered CRALBP.

In case of multi-objective algorithms in solving the ALBP, the most popular method utilized is the non-dominated sorting genetic algorithm II (NSGA-II) [23]. The other algorithms could be divided into two groups: local search-based multi-objective algorithms and swarm-based multi-objective algorithms. The swarm-based multi-objective algorithms is similar as the NSGA-II in handling the multiple objectives and they include the multi-objective artificial bee colony [24], Pareto grey wolf optimization [25] and several others. The local search-based multi-objective algorithms include the multi-objective simulated annealing algorithm [26], the iterated Pareto greedy algorithm [27] and others.

In case of RALBPs, the pioneering works are reported in Rubinovitz [28] and Rubinovitz et al. [29]. Since then, many optimization methods have been applied. Among the exact methods, the branch, bound and remember algorithm utilized by Borba et al. [30] is the best performer to solve the RALBP-II. The heuristic methods include cutting plane algorithm [31] and the beam search heuristics [32]. Researchers used metaheuristic algorithms and here a summary of several reported literatures on RALBP and its variants is presented. Levitin et al. [33] utilized genetic algorithm to solve RALBP-II and later Gao et al. [34] developed hybrid genetic algorithm and compared it with the published literature for solving RALBP-II. Daoud et al. [35] applied several hybrid methods to efficiently solve this problem. Particle swarm optimization (PSO) algorithms were utilized by Nilakantan et al. [36] to optimize cycle time and energy consumption in a robotic assembly line and later Nilakantan, Ponnambalam [37] applied PSO to minimize cycle time in a U-shaped robotic assembly line. Several bio-inspired search algorithms were utilized in Nilakantan et al. [38]. Differential evolution algorithm was used to minimize cycle time and cost by Nilakantan et al. [39]. Li et al. [40] used simulated annealing algorithms and recently Li et al. [2] and [12] applied migrating bird optimization algorithms to solve RALBP. There are also several more studies that reports the extension of RALBPs by considering various realistic objectives. Yoosefelahi et al. [41] studied the multiple objectives in the RALBP-II. Zhang et al. [24] and Zhou, Wu [42] applied metaheuristics to optimize energy consumption in RALBP. Minimizing the overall cost of performing the tasks were considered by Nilakantan et al. [39] and Pereira et al. [43]. Researchers have also worked on minimizing the carbon footprint for both straight [44] and U-shaped robotic assembly line [25]. Zhou, Wu [45] solved a bi-objective assembly line balancing problem considering time and space constraints. Studies have also considered specific attributes for RALBP and utilized metaheuristic approaches to solve them. RALBP with sequence dependent setup times was presented by Janardhanan et al. [12]. Two-sided RALBP using co-evolutionary PSO algorithm was reported by Li et al. [46] and discrete cuckoo search algorithms in Li et al. [20]. Rabbani et al. [47] solved RALBP with mixed-model for straight layout and Aghajani et al. [48] reported algorithms for solving mixed-model in two-sided assembly line layouts. Çil et al. [49] solved robotic parallel ALBP and researchers have reported several algorithms for solving U-shaped RALBP as seen in [25]. Li et al. [50] presented mathematical model and metaheuristics for simultaneous balancing and sequencing of robotic mixed-model assembly line.

Regarding the studies related to the human–robot collaboration in the assembly lines, Çil et al. [51] considers the semi-robotic ALBP and presents a mathematical model to solve one instance. Samouei, Ashayeri [52] considered the mixed-model ALBP with semi-automated operations and they present two models to tackle this problem. Weckenborg et al. [3] study the CRALBP to minimize the cycle time, where only one type of cobot is considered. They formulate this problem with a MILP model and present a hybrid genetic algorithm to tackle the large-size problems. Dalle Mura, Dini [53] present a genetic algorithm for the CRALBP, whereas the presented genetic algorithm is not a Pareto method. Yaphiar et al. [54] study the mixed-model ALBP with human–robot shared tasks and they present the MILP model to solve this problem. More recently, Rabbani et al. [55] take into account the mixed-model four-sided ALBP with the collaboration of human–robot and they present the augmented particle swarm optimization to tackle this problem, where the presented method is not a Pareto method. Table 1 summarizes all the relevant reported literature related to CRALBP.

From the detailed literature review, it can be seen that there is no study considering the purchasing costs of the cobots and there is no usage of Pareto approach for the multi-objective CRALBP. Hence, the proposed study presents a mathematical model and multi-objective migrating bird optimization algorithm to solve the cost-oriented CRALBP where several types of cobots are considered. Hence, this study is different from the reported literature in the context of problem setting and developed methodology.

3 Mathematical formulation

This section presents the considered problem and the formulated model.

3.1 Problem description

For the considered problem, cobots on the assembly line perform the tasks separately or in collaboration with the human workers. If one worker and one cobot are allocated to one station, there are three process alternatives: one task might be operated by a human worker, a cobot or the human worker and cobot in collaboration. As there are many types of cobot in real applications, the considered problem contains three sub-problems: (1) the task assignment; (2) the selection and the allocation of the human

Table 1 Summary of literature related to CRALBP

Papers (sorted by year)	Problem setting	Solution approaches
Çil et al. [51]	Single model semi-robotic ALBP	Mathematical model
Samouei, Ashayeri [52]	Mixed-model ALBP with semi-automated operations	Two mathematical models
Weckenborg et al. [3]	CRALBP where only one type of cobot is considered	Mathematical model and one objective genetic algorithm
Dalle Mura, Dini [53]	CRALBP where several types of cobots are considered	Genetic algorithm (not a Pareto method)
Yaphiar et al. [54]	Mixed-model ALBP with human–robot shared tasks	Mathematical model
Rabbani et al. [55]	Mixed-model four-sided ALBP with the collaboration of human–robot	Mathematical model and particle warm optimization algorithm (not a Pareto method)
Proposed study	Cost-oriented CRALBP where several types of cobots are considered	Mathematical model and multi-objective migrating bird optimization algorithm (Pareto method)

workers and cobots; (3) the selection of process alternative for the tasks.

Figure 2 provides an example layout of the assembly line with cobots. In this figure, four stations are opened and four workers and one cobot are allocated to the stations to operate the tasks. Especially, one worker and one cobot on station 4 operate task 8, task 10 and task 11 separately or collaboratively. Figure 3 illustrates the process methods of the tasks on station 3. As presented in the figure, it can be seen that task 8 and task 11 are operated by the worker and cobot in collaboration, whereas task 10 is operated by the worker alone.

On the basis of Weckenborg et al. [3], the main assumptions of this problem are given as follows. It is to be noted that this study does not consider the parallel working of a human worker and a cobot as presented in the seventh assumption. The parallel working might occur in some applications, however, in some occasions it is not allowed for the safety reasons. Specifically, when the parallel working of a human worker and a cobot occurs, the worker might concentrate on operating the tasks and collide with cobot due to lack of attention and hence the parallel working is not considered at all to avoid this possible danger. Still, the formulated model could be easily extended to consider the parallel working on the basis of Weckenborg et al. [3]. The assumptions of this study are listed as follows:

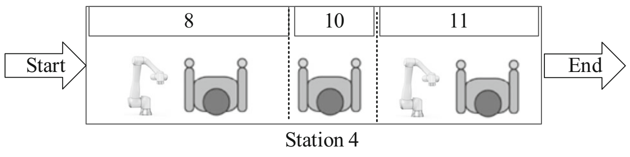

Fig. 3 Process method of the tasks on station 4

1. Only one single model product is assembled on the straight assembly line.
2. The operation times of tasks and the precedence relations are known in advance.
3. The operation/performance times of tasks depend on the process alternatives (human worker, cobots and in collaboration).
4. All the workers have the same skill set or can perform the tasks with the same operation time after receiving relevant short-time training.
5. There are several types of cobots available and each type of cobot might have different purchasing costs and different skills, namely different cobots might consume different times to complete the same task.
6. The cobots of each type are available without limitation, and it is permitted to allocate the cobots of each type to several stations.
7. The parallel working of a human worker and a cobot and is not allowed. Specifically, it is allowed to for the human worker or cobot to operate the tasks on one

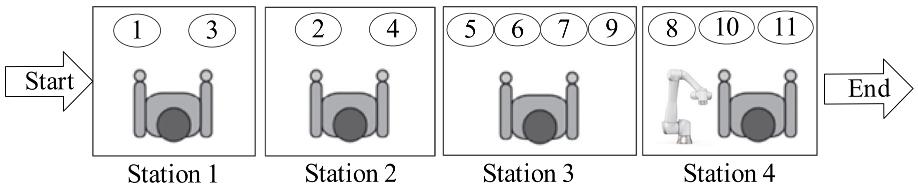

Fig. 2 Layout of assembly line with cobots

station in sequence in collaboration or not, but it is not permitted for the human worker and cobot to operate different tasks at the same time.

8. The loading and unloading times are neglected, and the maintenance operation is not considered.

3.2 Model formulation

Before presenting the formulated the model, the utilized indices and parameters are presented. It can be seen that this model utilizes the index r to denote the type of cobot, whereas Weckenborg et al. [3] only consider one type of cobot.

Indices and parameters	
i, j	Task index, $i,j \in \{1,2,\ldots,\mathrm{nt}\}$, where nt is the number of tasks
k, l	Station index, $k,l \in \{1,2,\ldots,\mathrm{ns}\}$, where ns is the number of stations
r	Cobot index, $r \in \{1,2,\ldots,\mathrm{nr}\}$, where nr is the types of available cobots
p	Process alternative index, $p \in \{1,2,\ldots,\mathrm{nr}+1,\mathrm{nr}+2,\ldots,2\cdot\mathrm{nr}+1\}$. Here, p = 1 when one task is operated by a human worker, $p = 1 + r$ when one task is operated by cobot r and $p = \mathrm{nr} + 1 + r$ when one task is operated by a worker and cobot r collaboratively
I	Set of tasks, $I = \{1,2,\ldots,\mathrm{nt}\}$
K	Set of stations, $K = \{1,2,\ldots,\mathrm{ns}\}$
R	Set of cobot types, ss
P	Set of process alternatives, $P = \{1,2,\ldots,\mathrm{nr}+1,\mathrm{nr}+2,\ldots,2\cdot\mathrm{nr}+1\}$
t_{ip}	Operation time of task i by processing alternative p
E	Set of pairs of tasks (i,j) in which task i is the immediate predecessor of task j
c_r	Purchasing cost of cobot r
ψ	A very larger positive number
wc_1, wc_2	Weighted costs
MB	Maximum budget to buy the cobots
MR	Maximum number of cobots to be allocated
MW	Maximum number of available workers
Decision variables	
CT	Cycle time
x_{ikp}	1, if task i is assigned to station k and operated by process alternative p; 0, otherwise
w_{rk}	1, if cobot r is allocated to station k
v_k	1, if a human worker is assigned to station k; 0, otherwise

This model is formulated based on the model presented in Weckenborg et al. [3]. The objective in expression (1) minimizes the cycle time; the objective in expression (2) minimizes the total cost of the cobots and workers, where the first part is the total purchasing cost of the cobots on all stations. Constraint (3) is the task occurrence constraint, indicating that each task must be allocated to one station and operated by one process alternative. Constraint (4) is the precedence constraint, indicating that the predecessors of one task must be allocated to the former station or the same station. Constraint (5) is the cycle time constraint, requesting that the total operation time of tasks by the corresponding process alternatives must be less than or equal to the cycle time. Constraint (6) and constraint (7) determine the value of w_{rk} Constraint (6) denotes that cobot r is allocated to station k when there is a task on station k that is operated by cobot r or a worker and cobot r collaboratively; constraint (7) denotes that there is at most one cobot on each station. Constraint (8) restricts that the number of the allocated cobots must be less than or equal to the limit, and it is disabled when there is no limit on the number of allocated cobots. Constraint (9) restricts that the total purchasing cost should not be larger than the maximum budget, and it is disabled when there is no limit on the budget or the budget is not available. Constraint (10) determines the assignment of workers and a worker is assigned to station k when there is a task on station k that is operated by a worker or a worker and cobot r collaboratively. Constraint (11) restricts that the number of available workers is not larger than the limit and it is disabled when there is no limit of the workers. Constraint (12) restricts the values of the decision variables.

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

$$\mathrm{Min}\, f_2 = \mathrm{wc}_1 \cdot \sum_{r\in R}\left(c_r \cdot \sum_{k\in K} w_{\mathrm{rk}}\right) + \mathrm{wc}_2 \cdot \sum_{k\in K} v_k \tag{2}$$

$$\sum_{k\in K}\sum_{p\in P} x_{\mathrm{ikp}} = 1 \;\forall i \in I \tag{3}$$

$$\sum_{k\in K}\sum_{p\in P} k \cdot x_{\mathrm{ikp}} - \sum_{l\in K}\sum_{p\in P} l \cdot x_{\mathrm{jlp}} \le 0 \;\forall (i,j) \in E \tag{4}$$

$$\sum_{i\in I}\sum_{p\in P} t_{\mathrm{ip}} \cdot x_{\mathrm{ikp}} \le \mathrm{CT} \;\forall k \in K \tag{5}$$

$$\psi \cdot w_{\mathrm{rk}} \ge \sum_{i\in I}\left(x_{\mathrm{ik},1+r} + x_{\mathrm{ik},\mathrm{nr}+1+r}\right) \;\forall k \in K, r \in R \tag{6}$$

$$\sum_{r\in R} w_{rk} \le 1 \;\forall k \in K \tag{7}$$

$$\sum_{r\in R}\sum_{k\in k} w_{\mathrm{rk}} \le \mathrm{MR} \tag{8}$$

$$\sum_{r\in R}\left(c_r\cdot\sum_{k\in K}w_{\mathrm{rk}}\right)\leq \mathrm{MB} \tag{9}$$

$$\psi\cdot v_k\geq\sum_{i\in I}x_{\mathrm{ik},1}+\sum_{i\in I}\sum_{p\in\{\mathrm{nr}+2,\ldots,2\cdot\mathrm{nr}+1\}}x_{\mathrm{ikp}}\ \forall k\in K \tag{10}$$

$$\sum_{k\in K}v_k\leq \mathrm{MW} \tag{11}$$

$$x_{\mathrm{ikp}},w_{\mathrm{rk}},v_k\in\{0,1\} \tag{12}$$

This formulated model is a MILP model, and it can be solved utilizing the CPLEX solver with either objective or weighted objective in expression (13). Here, w_1 and w_2 are weights, $f_1^{\min}$ and $f_1^{\max}$ are the minimum value and maximum value of f_1, and $f_2^{\min}$ and $f_2^{\max}$ are the minimum value of maximum value of f_2. After testing many combinations of the w_1 and w_2, the CPLEX solve can achieve a set of Pareto solutions.

$$\mathrm{Min}f=w_1\cdot\left(f_1-f_1^{\min}\right)/\left(f_1^{\max}-f_1^{\min}\right)+w_2\cdot\left(f_2-f_2^{\min}\right)/\left(f_2^{\max}-f_2^{\min}\right) \tag{13}$$

4 Proposed multi-objective algorithms

MBO has been applied in many single objective assembly line optimization problems [2, 11, 12] successfully. The multi-objective migrating bird optimization has also been applied to solve multi-objective flowshop scheduling problems and it has shown good performance in solving them [56]. In addition, MBO can be utilized to solve the discrete optimization problem directly utilizing the neighbour operator. However, there is no application of this method in solving the multi-objective CRALBP. Hence, this study presents the multi-objective migrating bird optimization (MMBO) algorithm to solve the considered multi-objective problem effectively and presents several modifications to enhance the performance of the algorithm. 1) MMBO algorithm utilizes new population selection and update on the basis of the fast non-dominated sorting approach; 2) this algorithm utilizes the restart mechanism to select one solution in the in the permanent Pareto archive to replace the abandoned solution which remains unchanged for several iterations. The main procedure and the components of the developed MMBO are described in the following sections as follows.

4.1 Original migrating bird optimization

MBO is designed on the basis of the V flight formation of the migrating birds. There is a leader bird leading the flock and the remained birds follow the leader in the left and right side to form the V-shape [11]. MBO is a population algorithm based on local search and the main procedure of the MBO is presented in Algorithm 1. The algorithm has four parameters and they are: the population size (n), the number of neighbour solutions (k), the number of neighbour solutions to be shared with the next individual (x) and the number of tours before replacing the leader (m). After initializing the population, MBO conducts the leader improvement, block improvement and leader replacement until the termination criterion are met. In the leader improvement phase, the leader tries to improve itself by generating and evaluating k neighbour solutions. In the block improvement phase, each individual is attempted to be improved by evaluating its (k-x) neighbour solutions and x unused best neighbour solutions in the front. In the leader replacement phase, the leading individual is removed to the end of either side and the following individual is forwarded to the leading position after executing the leader improvement and block improvement for m iterations.

The benefit mechanism is the main feature which differentiates the MBO from other population algorithms. The benefit mechanism helps to accelerate the evolution process by replacing the poor-quality solutions with the neighbour solutions of the better individuals.

Algorithm 1: Main procedure of the MBO algorithm

```
Input: Algorithm parameters and instance data
% Initialization
Initialize the $n$ solutions randomly and place the population in the V-shape;
Do
 For $i$=1 to $m$ do
  % Leader improvement
  Obtain $k$ neighbor solutions of the leader individual;
  Update the leader individual with the best neighbor solution when the best
  neighbor solution achieves better fitness;
  % Block improvement
  For each individual in the left and right sides
  Obtain ($k$-$x$) neighbor solutions of this individual;
  Update this individual with the best one of the its ($k$-$x$) neighbor solutions and
  $x$ unused best neighbor solutions in the front when the best neighbor solution
  achieves better fitness;   %Benefit mechanism
  Endfor
 Endfor
 % Leader replacement
 Remove the leading individual to the end of either side;
 Forward the following individual to the leading position as the new leader;
Until (termination criterion is satisfied)
Output: Best objective value
```

4.2 Encoding and decoding

As the considered problem involves the task assignment and the selection and the allocation of the human workers and cobots, this study utilizes task permutation vector and process alternative selection vector for encoding. Task permutation vector is the sequences of all tasks and the tasks in the former positions have higher priorities and should be allocated at first. Process alternative selection vector corresponds to the process alternatives on stations, e.g. code p in position k donates that process alternative p is allocated to station k. An example of encoding scheme is presented in Fig. 4, where four types of cobots are available.

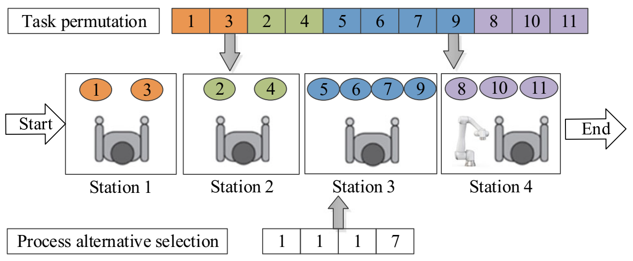

Fig. 4 Illustrated encoding and decoding

As this study considers type II CRALBP with two objectives, it is necessary to determine the appropriate cycle time for each individual. On the basis of Zhang et al. [24], the first method for decoding is presented in Algorithm 2 and Algorithm 3. Namely, the CT_{LB} is firstly tested as the initial cycle time CT_{Init} and CT_{Init} increases by $CT_{Init} \leftarrow CT_{Init} + 1$ until a feasible solution is obtained. In the decoding procedure, the allocation of the human workers and cobots is determined directly by the process alternative selection vector at first. Afterwards, as more tasks are possible to be assigned to station k and one task is assignable when 1) its predecessors have been assigned to the former station or the former positions of station k; 2) it can be completed within CT_{Init} by any process alternatives available on station k when $k < ns$. Here, the task is operated by the process alternative which completes the task with the smallest operation time when there are several

```
Algorithm 2: Procedure to determine the best cycle time with increments
Step 1: Obtain the lower bound of the cycle time $CT_{LB}$ and set $CT_{LB}$ as the initial cycle time $CT_{Init}$, where $CT_{LB} = \left(\sum_i \min_p t_{ip}\right)/ns$.
Step 2: Decode with $CT_{Init}$ as the initial cycle time and achieve the corresponding objective values;
Step 3: Terminate this procedure and output the objective values when the achieved CT is not larger than the $CT_{Init}$;
Step 4: Update $CT_{Init} \leftarrow CT_{Init} + 1$ and execute Step 2;
```

process alternatives available. Figure 4 provides an illustrated example for the decoding procedure.

```
Algorithm 3: Decoding procedure with $CT_{Init}$ as the initial cycle time
Determine the allocation of the human workers and cobots based on the process alternative selection vector;
For $k := 1$ to $ns$
  While (1)
    Determine the assignable task set;
    % One task is added to the assignable task set only if 1) its predecessors have been assigned to the former station or the former positions of station $k$; 2) it can be completed within $CT_{Init}$ by any process alternatives available on station $k$ when $k < ns$;
    Terminate this while loop when there is no assignable task;
    Select the task in the assignable task set based on the task permutation;
    Select the process alternative which completes the task with the smallest operation time and set the corresponding time as the operation time of this tasks;
    Assign this task to station $k$ and reduce the remained capacity of station $k$;
  Endwhile
Endfor
```

However, this procedure to determine the best cycle time in Algorithm 2 might be very slow in solving the large-size instances. In fact, it is necessary to conduct the decoding procedure in Algorithm 3 for several times to obtain the proper cycle time, leading to a lot of time wastage. Hence, this study proposes a method based on dichotomy to determine the best cycle time in Algorithm 4. The preliminary experiments show that the new method based on dichotomy produces superior performance in solving the large-size instances and hence it is proposed in this study.

Algorithm 4: Procedure to determine the best cycle time based on dichotomy

```
Step 1: Obtain the lower bound of the cycle time $CT_{LB}$ and set $CT_{Start} \leftarrow CT_{LB}$, $CT_{End} \leftarrow 2 \cdot CT_{LB}$ and $CT_{Gap} \leftarrow \lceil (CT_{End} - CT_{Start})/2 \rceil$;
Step 2: Execute Step 4 when $CT_{Start} + 2 \geq CT_{End}$; otherwise, decode with $CT_{End}$ as the initial cycle time and achieve the corresponding objective values;
Step 3: If the achieved CT is not larger than $CT_{End}$, update $CT_{End} \leftarrow CT_{End} - CT_{Gap}$ and $CT_{Gap} \leftarrow \lceil (CT_{End} - CT_{Start})/2 \rceil$; otherwise, update $CT_{End} \leftarrow CT_{End} + CT_{Gap}$, $CT_{Start} \leftarrow CT$ and $CT_{Gap} \leftarrow \lceil (CT_{End} - CT_{Start})/2 \rceil$. Afterwards, execute Step 2.
Step 4: Set $CT_{Start}$ as the initial cycle time $CT_{Init}$。
Step 5: Decode with $CT_{Init}$ as the cycle time and achieve the corresponding objective values;
Step 6: Terminate this procedure and output the objective values when the achieved CT is not larger than the $CT_{Init}$; otherwise, execute Step 7.
Step 7: Update $CT_{Init} \leftarrow CT_{Init} + 1$ and execute Step 5;
```

4.3 Multi-objective migrating bird optimization

On the basis of the fast non-dominated sorting approach by Deb et al. [23], the main procedure of the MMBO is illustrated in Algorithm 5. MMBO starts with initializing *n* solutions randomly and the main loop is repeated until a termination criterion is met. Within the loop, the leader obtains *k* neighbour solutions and each remained individual obtains (*k*-*x*) neighbour solutions. Afterwards, the best *n* individuals are selected from the original population and all the neighbour solutions ($k + (n-1) \cdot (k-x) + n$ in total) as the new population based on the fast non-dominated sorting approach. At last, this algorithm utilizes the restart mechanism to select one solution in the permanent Pareto archive to replace the abandoned solution which remains unchanged for several iterations. It can be noted that restart mechanism rather than leader replacement is utilized here to emphasize exploration. The fast non-dominated sorting approach and the restart mechanism will be explained later in detail.

Algorithm 5: Main procedure of the MMBO algorithm

```
Input: Algorithm parameters and instance data (precedence relation and task operation times)
% Initialization
Initialize the $n$ solutions randomly;
Do
    % Leader improvement
    Obtain $k$ neighbor solutions of the leader individual and replace the leader individual
    immediately once the neighbor solution dominates or equal to the leader individual;
    % Block improvement
    For each individual in the left and right sides
        Obtain ($k$-$x$) neighbor solutions of this individual and replace the incumbent individual
        immediately once the neighbor solution dominates or equal to the incumbent individual;
    Endfor
    Combine the original population and all the neighbor solutions ($k + (n-1) \cdot (k-x) + n$ in
    total) and remove the individuals with the same objective function values;
    Calculate the ranks and crowding distances of all individuals based on fast non-dominated
    sorting approach;
    Sort individuals based on ranks and crowding distances, and select the best $n$ individuals to
    form the new population;
    % Restart mechanism
    For any individual that that has not been updated in consecutive $\alpha$ iterations
        Conduct a restart mechanism to select a solution from the permanent Pareto frontier or
        generate a solution randomly to replace the individual;
    Endif
Until (termination criterion is satisfied)
Output: A set of Pareto solutions
```

During the population selection, the fast non-dominated sorting approach is presented here to achieve the ranks and crowding distances. Namely, the proposed MMBO select the best n individuals with the smaller ranks or the larger crowding distances when the ranks are the same from the original population and all the neighbour solutions.

```
Population selection based on the fast non-dominated sorting approach
Input: Original population and all the neighbor solutions
% Calculation of the ranks
Step 1: Determine the domination relations between the individuals, where individual $a$ dominates
individual $b$ ($a \prec b$) when solution $a$ is no worse than solution $b$ in any objective and objective $b$ is
worse than solution $a$ in at least one objective. Remove all the non-dominated solutions to set $F_1$ (the
rank is 1), and subsequently select and remove the remained non-dominated solutions to set $F_2$ until
obtaining the ranks of all the individuals.
% Calculation of crowding distances
Step 2: For the individuals of the same rank, the crowding distances are calculated as follows.
Here, $POS$ is the set of the individuals of the same rank, $Obj_m$ is the objective value of the
objective $m$ and $+\infty$ is a very large positive number.
$Setsize \coloneqq |POS|$;
For all $m$ objectives
  Sort the Pareto front solutions in the increasing order of the objective $m$;
  $f_m^{max} \coloneqq$ Maximum value of objective $m$;
  $f_m^{min} \coloneqq$ Minimum value of objective $m$;
  For all $i$:=2,…,$Setsize$-1
    $POS[i]_{dist} := POS[i]_{dist} + (POS[i+1].Obj_m - POS[i-1].Obj_m)/(f_m^{max} - f_m^{min})$;
  Endfor
Endfor
$POS[1]_{dist} \coloneqq +\infty$ ; $POS[Setsize]_{dist} \coloneqq +\infty$ ;
% Population selection
Step 3: Sort the individuals in the increasing order of the ranks or sort the individuals in the decreasing
order of the crowding distance when the ranks are the same. Afterwards, select the former $n$ individuals
as the new population;
Output: The best $n$ individuals as the new population
```

When utilizing the population selection based on the fast non-dominated sorting approach, the isolated solution will be preserved for many times, leading to possibly being trapped into local optima. Hence, this study presents the restart mechanism based on the modified crowding distance to select one solution in the permanent Pareto archive to replace the abandoned solution. The developed restart mechanism based on the modified crowding distance is illustrated as follows. In the modified crowding distance, the crowding distances of the extreme solutions are set to 1.0 rather than $+\infty$. And the number of the times of being selected is also considered utilizing the

$$\mathrm{POS}[i]_{\mathrm{dist}} \cdot \mathrm{pow}\left(0.5, \mathrm{POS}[i]_{\mathrm{Selectcounter}}\right)$$

and hence the solutions will not be selected again if it has been selected for several times. These modifications ensure that the isolated solutions are first selected and also to avoid the situation of selecting the same extreme or isolated solutions repeatedly. In fact, after conducting the restart mechanism for many times, all the Pareto solutions will be selected and tested to enhance the exploration capacity. Meanwhile this study utilizes the neighbour solution of the selected one to replace the abandoned solution if the solution in the permanent Pareto archive has been selected for several times. The utilization of the neighbour solution further enhances the exploration capacity and brings in new and possibly high-quality solutions to the population to escape from getting trapped into local optima. To further enhance the exploration capacity, when the selected individual has been selected for many times (set to 100), this study replaces the abandoned solution with a randomly generated solution.

```
Restart mechanism based on the modified crowding distance
$Setsize := |POS|$; % $POS$ is the set of the individuals
For all $m$ objectives
    Sort the Pareto front solutions in the increasing order of the objective $m$;
    $f_m^{max} :=$ Maximum value of objective $m$;
    $f_m^{min} :=$ Minimum value of objective $m$;
    For all $i$:=2,…,$Setsize$-1
        $POS[i]_{dist} := POS[i]_{dist} + (POS[i+1].Obj_m - POS[i-1].Obj_m)/(f_m^{max} - f_m^{min})$;
    Endfor
Endfor
$POS[1]_{dist} := 1.0$ ; $POS[Setsize]_{dist} := 1.0$ ;
For all $i$:=1,…,$Setsize$
$POS[i]_{dist} := POS[i]_{dist} \cdot pow(0.5, POS[i]_{Selectcounter})$;
% $POS[i]_{Selectcounter}$ is the number of times that individual $i$ has been selected;
Endfor
Select the individual with the largest value of $POS[i]_{dist}$ and 1) utilize the selected individual to replace the abandoned solution when $POS[i]_{Selectcounter} < 20$; 2) or utilize the neighbor solution of the selected one to replace the abandoned solution when $POS[i]_{Selectcounter} < 100$ ; 3) or generates a solution randomly to replace the abandoned solution when $POS[i]_{Selectcounter} \geq 100$.
```

To obtain high-quality neighbour solutions in the leader improvement and block improvement phases, this study utilizes the neighbour structures presented in Fig. 5. As it can be seen, swap operator and insert operator are utilized to modify the task permutation vector. The swap operator selects two tasks in different positions randomly and exchanges their positions; the insert operator selects one task randomly and inserts it into a different position. For the process alternative selection vector, the swap operator and mutation operator are utilized. The swap operator selects two different positions randomly and exchanges the possess alternatives on the two positions; the mutation operator selects one position randomly and replaces the possess alternative on this position with a different possess alternative. As two vectors need to be optimized, this study first selects one vector randomly and later selects one neighbour structure randomly to modify the selected vector and obtain a new vector.

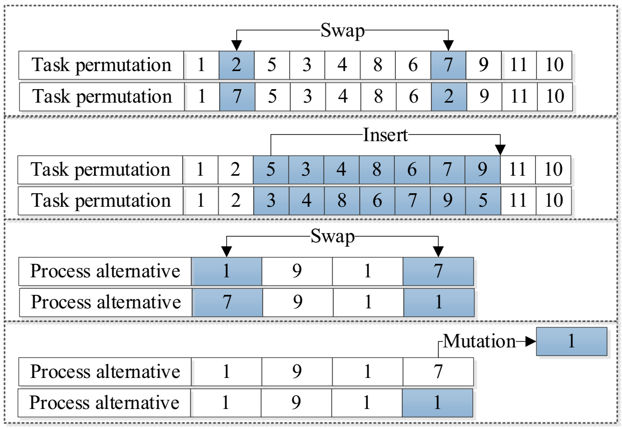
Swap
Task permutation 1 2 5 3 4 8 6 7 9 11 10
Task permutation 1 7 5 3 4 8 6 2 9 11 10
Insert
Task permutation 1 2 5 3 4 8 6 7 9 11 10
Task permutation 1 2 3 4 8 6 7 9 5 11 10
Swap
Process alternative 1 9 1 7
Process alternative 7 9 1 1
Mutation 1
Process alternative 1 9 1 7
Process alternative 1 9 1 1

Fig. 5 The utilized neighbour structure

4.4 An illustrated example

This section presents an illustrated example with 11 tasks and four stations and there are four types of cobots (cobot 1, cobot 2, cobot 3 and cobot 4) available. The precedence relations and the operation times of tasks by process alternatives are presented in Table 2. Among the process alternatives, the number 1 corresponds to operating tasks by a human worker; the number 2, number 3, number 4 and number 5 correspond to operating tasks by cobot 1, cobot 2, cobot 3 and cobot 4; the number 6, number 7, number 8 and number 9 correspond to operating tasks by worker and

Table 2 The precedence relations and the operation times of tasks

Task	Successors	Operation times by process alternatives								
		1	2	3	4	5	6	7	8	9
1	2, 3	4	–	7	–	–	–	3	–	–
2	4	5	–	–	8	–	–	4	3	3
3	5	5	–	–	–	–	–	–	3	3
4	6	6	–	11	–	–	–	4	–	–
5	7, 8, 9	3	–	–	–	–	–	–	–	2
6	10	2	4	–	3	–	2	–	2	2
7	11	1	–	–	–	–	–	–	–	–
8	11	7	–	–	–	–	–	5	5	–
9	11	5	–	–	–	–	4	–	3	–
10	11	2	–	–	–	–	2	–	–	–
11	–	6	–	11	9	8	–	4	4	4

cobot 1 in collaboration, worker and cobot 2 in collaboration, worker and cobot 3 in collaboration, and worker and cobot 4 in collaboration. And the purchasing costs of the four cobots are set as 10.11, 12.79, 18.55 and 20.83; respectively.

Table 3 illustrates the Pareto solutions by the proposed method when wc_1 and wc_2 are set as 1.0 and 0.0. As you see, the cycle time can be reduced with the increased cost of cobots. For instance, if no cobot is utilized, the achieved cycle time is 12; if the cobot 2 is utilized, the cycle time is reduced to 11. And Fig. 6 illustrates the detailed task assignment and worker and cobot allocation for the solutions with a cycle time of 12 (Fig. 6a) and a cycle time of 11 (Fig. 6b).

5 Experimental tests and results

This section presents the results obtained by testing the formulated model and evaluating the performance of the MMBO algorithm in comparison with several other multi-objective algorithms.

5.1 Experimental settings

As there are no published studies solving the considered problem, this study generates a set of instances as follows.

Table 3 Pareto solutions by the proposed method

Pareto solution	f_1	f_2
1	9	44.13
2	10	25.58
3	11	12.79
4	12	0

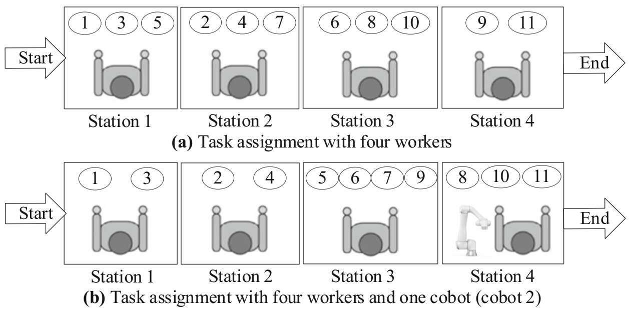

Fig. 6 Detailed task assignment and worker and cobot allocation

Firstly, this study selects 22 precedence diagrams, each with several number of stations, from the Scholl's instances (see http://www.assembly-line-balancing.de). Secondly, this study sets that the workers are available without limitation (wc_1 and wc_2 are set as 1.0 and 0.0) and there are four types of cobots available. And the cost of cobot r ($r \in \{1, 2, \ldots, 4\}$) is set as $c_r = 6 + 4 \cdot r \cdot \text{Rand}[0.8,\ 1.2]$, where Rand $[0.8,\ 1.2]$ is a random number within $[0.8,\ 1.2]$. Thirdly, the operation times of tasks by process alternatives are generated: $t_{i,1} = t_i$, $t_{i,1+r} = t_i \cdot (2 - 0.25 * (r-1))$, $t_{i,\,\text{nr}+1+r} = t_i \cdot (0.7 - 0.05 * (r-1))$, where t_i is the original operation times taken from the literature. Namely, the operation times by the cobot are set larger than that by the worker and the operations times by the collaboration are set smaller than that by the worker. As the cobots have limited capacities, this study also sets that the cobots might not be capable of operating some tasks. Specifically, it is set that cobot r cannot operate task i when $\text{Rand}[0.0,\ 1.0] < 0.8 - 0.02 \cdot (r-1)$ and the collaboration of worker and cobot r cannot operate task i when cobot r cannot operate task i and

$$\text{Rand}[0.0, 1.0] < (0.6 - 0.05 \cdot (r-1))/(0.8 - 0.02 \cdot (r-1))$$

, where Rand $[0.0,\ 1.0]$ is a random number within $[0.0,\ 1.0]$. It is to be noted that for the data generated it is set here that the expensive cobots have larger capacities and consumes less times to complete tasks separately or in collaboration. In total, there are 93 instances, ranging from the smallest-size instance with 7 tasks to the largest-size instance with 297 tasks. The cobot costs, precedence relations and tasks operation times of the generated instances, are available upon request.

To test the formulated model, this study solves one instance as an example, where the 101 combinations of the w_1 and w_2 ($w_1 = 1 - w_2$) in expression (13) are tested. To investigate the main segments of the proposed MMBO, the proposed MMBO is compared with three original multi-objective migrating bird optimization algorithms: 1) OMMBO1: in the leader and block improvement, the neighbour solution replaces the incumbent individual immediately once the neighbour solution dominates or equal to the incumbent individual, this is not applied in this algorithm and the restart mechanism is also not applied. 2) OMMBO2: the restart mechanism is not applied. 3) OMMBO3: in the leader and block improvement, the neighbour solution replaces the incumbent individual immediately once the neighbour solution dominates or equal to the incumbent individual, this is not applied in this algorithm. To evaluate the developed algorithm, the proposed MMBO is compared with five other algorithms: multi-objective simulated annealing algorithm (MSA) [57], non-dominated sorting genetic algorithm II (NSGA-II) [23], original multi-objective artificial bee colony (OMABC) algorithm, improved multi-objective artificial bee colony algorithm (IMABC) [24]. All the compared algorithms are re-implemented utilizing the same encoding, decoding and neighbour operators. All the implemented algorithms solve all the instances for 10 repetitions under the termination criterion of an elapsed computation time of $\mathrm{nt} \cdot \mathrm{nt} \cdot \tau$ milliseconds. Here, this value of τ is set to 10, 20 and 30 to observe the performances of algorithms from short to large computation time.

The CPLEX solver of General Algebraic Modelling System 23.0 is utilized to solve the mathematical model. And the CPLEX solver terminates when the optimal solution is achieved and verified. The implemented algorithms are coded with the C + + programming language of the platform of Microsoft Visual Studio 2015. The model in the CPLEX solver and the main procedures of the algorithms are available upon request. All the experiments are running on a set of virtual computers, equipped with one processor and 2 GB RAM memory, of a tower type of server. This tower type of server has two Intel Xeon E5-2680 v2 processors and 64 GB RAM memory.

5.2 Evaluation indicators

The evaluation of single objective is straightforward, whereas the evaluation of conflicted multiple objectives might be difficult and sometimes ambiguous. This study selects four quantitative indicators and one graphical indicator to evaluate the tested algorithms. The four quantitative indicators are the convergence of the Pareto-optimal solution (CP), the spread metric (SP), hyper volume ratio (HVR) and Unary Epsilon Indicator (I_ε^1). Notice that HVR and I_ε^1 are Pareto compliant indicators and the HVR might be regarded as the most appropriate scalar indicator as it considers both the convergence to the true frontier and the spread of the achieved Pareto solutions [44]. Although the CP and SP are not Pareto compliant indicators, they are also selected in this study as they have been widely utilized in the literature. Hence, this study sets that the HVR has the highest priority and I_ε^1 has the second-highest priority.

The indicator CP measures the difference between the achieved Pareto archive S and the true Pareto archive P, which is calculated in expression (14) and expression (15). Here, $\mathrm{d}t_i$ is the smallest distance between the achieved Pareto solution i and the true Pareto archive P. And the smaller value of CP donates the better convergence to the true Pareto archive. The indicator SP measures the spread of the achieved Pareto archive S utilizing expression (16) and expression (17). Here, sd_{f1} and sd_{f2} are the Euclidean distances between the achieved extreme solutions and the true extreme solutions. After sorting the achieved Pareto solutions in the increasing order of the first objective, the Euclidean distance between adjacent individuals sd_i is calculated with expression (17) and $\overline{\mathrm{sd}}$ is the average value of the Euclidean distances of all individuals. The small value of SP means the better spread of the achieved Pareto archive.

$$\mathrm{CP} = \sum_{i=1}^{|s|} \mathrm{d}t_i / |S| \tag{14}$$

$$\mathrm{d}t_i = \min_{n=1,\cdots,|P|} \sqrt{\sum_{m=1}^{2} \left[\frac{f_m(x_i) - f_m(y_n)}{f_m^{\max} - f_m^{\min}} \right]^2} \tag{15}$$

$$\mathrm{SP} = \frac{\mathrm{sd}_{f1} + \mathrm{sd}_{f2} + \sum_{i=1}^{|S|-1} |\mathrm{sd}_i - \overline{\mathrm{sd}}|}{\mathrm{sd}_{f1} + \mathrm{sd}_{f2} + (|S|-1) \cdot \overline{\mathrm{sd}}} \tag{16}$$

$$\mathrm{sd}_i = \sqrt{\sum_{m=1}^{2} \left[\frac{f_m(x_i) - f_m(x_{i+1})}{f_m^{\max} - f_m^{\min}} \right]^2} \tag{17}$$

For the two Pareto compliant indicators, the HVR is the ratio of the hyper volume of the achieved Pareto archive S and that of the true Pareto archive P utilizing expression (18) [58]. The value of HVR is less than or equal to 1.0 and a value closed to 1.0 means that the achieved Pareto archive is very closed to the true Pareto archive. And the larger value of HVR donates the superior Pareto archive. The I_ε^1 is the minimum distance between the achieved Pareto archive S and that of the true Pareto archive utilizing expression (19) [58]. The value of I_ε^1 is larger than or equal to 1.0 and the achieved Pareto archive is very closed to the true Pareto archive when the value of I_ε^1 is closed to 1.0. In addition, the smaller value of HVR denotes the superior Pareto archive.

$$\mathrm{HVR} = \frac{\mathrm{volume}\left(\bigcup_{i=1}^{s} x_i\right)}{\mathrm{volume}\left(\bigcup_{j=1}^{p} x_j\right)} \tag{18}$$

$$I_\varepsilon^1 = I_\varepsilon(S, P) = \max_{x^2} \min_{x^1} \min_{j} \frac{f_j(x^1)}{f_j(x^2)} \tag{19}$$

All the quantitative indicators mentioned cannot provide information the spatial behavior of the implemented methods in the objective space, and hence this study utilizes the empirical attainment function (EAF) [58], a graphical indicator, to have a better observation of the algorithms. The EAF is calculated using expression (20) to expression (30), where nh is the number of repetitions and Γ_h is the achieved Pareto archive in one independent run by algorithm α. Namely, the EAF is achieved by running the algorithms for many times to empirically approximate the probability. To evaluate the difference between algorithms, the difference between two EAFs (donated as DEAF) is calculated in expression (22). For algorithm α and algorithm β, algorithm α has the larger probability to obtain the Pareto archive to dominate the point ω when DEAF in this point is larger than 0.0; otherwise, algorithm β has the larger probability to obtain the Pareto archive to dominate this point.

$$\mathrm{EAF}_{\alpha}(\omega) = \frac{1}{\mathrm{nh}} \sum_{h=1}^{\mathrm{nh}} I\left(\Gamma_h^{\alpha} \trianglelefteq \omega\right) \tag{20}$$

$$I\left(\Gamma_h^{\alpha} \trianglelefteq \omega\right) = \begin{cases} 1 & \text{if } \Gamma_h^{\alpha} \trianglelefteq \omega \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

$$\mathrm{DEAF}_{\alpha,\beta}(\omega) = \frac{1}{\mathrm{nh}} \sum_{h=1}^{\mathrm{nh}} \left[I\left(\Gamma_h^{\alpha} \trianglelefteq \omega\right) - I\left(\Gamma_h^{\beta} \trianglelefteq \omega\right) \right] \tag{22}$$

5.3 Comparative results

As the true Pareto archive is unknown, this study runs all the algorithms for ten times with $\tau = 100$ and the achieve Pareto archive by all the algorithm is regarded as the nearly true Pareto archive. Before conducting the experiments, the parameters of the implemented algorithms are first calibrated with the full factorial design of experiments. Specifically, ten different instances from both small and large-sized instances are selected and solved for ten repetitions by all the possible combinations of the parameter values. Then, multi-factor analysis of variance (ANOVA) technique is utilized to select the best parameter values, where the average value of 1-HVR in one run is the response variable. Detailed parameter calibration is not presented due to space reasons, the authors can refer to Nilakantan et al. [44] and Ciavotta et al. [58] for the detailed descriptions of this approach.

This study first evaluates the formulated model by solving the instance illustrated in Sect. 4.4. For this instance, a total number of 101 combinations of w_1, and w_2 are tested ($w_2 = 1 - w_1$) and the w_1 is set as $0.00, 0.01, \ldots, 0.99, 1.00$; respectively. Table 4 presents the result by CPLEX solver and Pareto solutions by proposed MMBO with $\tau = 10$ in ten runs. It is observed that the CPLEX solver is capable of achieving five different solutions and three Pareto solutions (two solutions are dominated) by testing the 101 combinations of the w_1 and w_2. The proposed MMBO is capable of achieving four Pareto solutions and all the Pareto solutions obtained by the CPLEX solver belongs to the Pareto archive achieved by the MMBO. The solutions obtained using MMBO outperforms the CPLEX solver in solving the considered problem by achieving one more solution than the CPLEX solver. Another drawback of utilizing the CPLEX solver is that it is necessary to test many combinations of the w_1 and w_2, leading to unacceptable computation time in solving the large-size instances.

For the situation when $w_1 = 1.00$, it is clear that the sole optimization of cycle time might lead to a large purchasing cost; for the situation when $w_1 = 0.00$, it is observed that sole optimization of purchasing cost might lead to a large cycle time. This finding suggests that the utilization of multi-objective is reasonable.

To evaluate the segments of the proposed MMBO, Table 5 presents the results by the OMMBO1, OMMBO2, OMMBO3 and MMBO in terms of HVR and I_{ε}^{1}. As HVR and I_{ε}^{1} are Pareto compliant indicators and have the highest priority, the results in terms of *CP* and *SP* are not presented here, and they are available upon request. As seen from the results, MMBO outperforms OMMBO1 and OMMBO2 clearly, demonstrating the restart mechanism is quite effective. MMBO and OMMBO3 obtains similar performance, but still MMBO outperforms OMMBO3 slightly in terms of the HVR. In short, this comparative study demonstrates that the restart mechanism enhances the performance of the proposed MMBO algorithm by a significant margin.

To evaluate the performance of the implemented algorithms, Table 6 exhibits the results by all the implemented algorithm in terms of HVR and I_{ε}^{1} when $\tau = 30$. It is observed that MMBO is the best performer regarding the HVR, OMABC is the second-best performer and MSA is the worst performer. Regarding the I_{ε}^{1}, the MMBO is also the best performer, OMABC is the second-best performer and the MSA is the worst performer. In summary, the developed MMBO obtains superior performance than all the other implemented algorithm in terms of HVR and I_{ε}^{1}. It is to be noted that based on Salehi et al. [59], the total time complexities of MSA, OMABC, NSGA-II, IMABC and MMBO are $O(N^5)$, $O(N^5)$, $O(N^5)$, $O(N^5)$ and $O(N^6)$; respectively. MMBO has the largest time complexity. However, all the algorithms utilize the same computation time as the termination criterion and they have utilized the same decoding procedures.

Table 4 Results by CPLEX solver and proposed MMBO

Number	CPLEX solver				Proposed MMBO			
	f_1	f_2	w_1	CPU(s)	Number	f_1	f_2	CPU(s)
1	9	62.68	1.00	0.49	1	9	44.13	1.10
2	9	44.13	0.56–0.99	0.45	2	10	25.58	1.10
3	10	25.58	0.47–0.55	0.57	3	11	12.79	1.10
4	12	0.00	0.01–0.46	0.39	4	12	0	1.10
5	40	0.00	0.00	0.27				

Table 5 Results by the MMBO algorithms in terms of HVR and I_ε^1 when $\tau = 30$

Instance	HVR				I_ε^1			
	OMMBO1	OMMBO2	OMMBO3	MMBO	OMMBO1	OMMBO2	OMMBO3	MMBO
P7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
P8	1.000	1.000	1.000	1.000	1.004	1.000	1.000	1.000
P9	1.000	1.000	1.000	1.000	1.001	1.000	1.000	1.000
P11	0.994	0.998	0.997	0.998	1.063	1.008	1.029	1.017
P21	0.983	0.987	0.997	0.995	1.176	1.132	1.021	1.036
P25	0.984	0.981	0.990	0.990	1.109	1.130	1.057	1.060
P28	0.983	0.983	0.991	0.992	1.069	1.068	1.034	1.033
P29	0.958	0.962	0.979	0.978	1.256	1.228	1.097	1.116
P30	0.968	0.966	0.977	0.979	1.176	1.179	1.117	1.105
P32	0.978	0.981	0.990	0.990	1.088	1.068	1.041	1.043
P35	0.979	0.981	0.991	0.990	1.206	1.197	1.082	1.098
P45	0.983	0.984	0.989	0.989	1.081	1.075	1.044	1.045
P53	0.999	0.999	1.000	1.000	1.012	1.012	1.005	1.006
P58	0.962	0.963	0.975	0.976	1.222	1.217	1.090	1.090
P70	0.969	0.967	0.975	0.976	1.118	1.133	1.075	1.075
P75	0.964	0.966	0.971	0.971	1.187	1.170	1.102	1.100
P83	0.971	0.970	0.976	0.975	1.113	1.122	1.085	1.094
P89	0.965	0.968	0.965	0.965	1.367	1.341	1.230	1.217
P94	0.974	0.971	0.979	0.982	1.115	1.137	1.073	1.063
P111	0.966	0.966	0.974	0.974	1.108	1.105	1.070	1.073
P148	0.975	0.972	0.984	0.985	1.074	1.086	1.042	1.043
P297	0.961	0.962	0.971	0.971	1.113	1.103	1.084	1.085
Avg	0.978	0.979	0.985	**0.986**	1.123	1.115	1.062	1.063

*Best in bold

To ascertain that the proposed algorithm is indeed more effective and observed difference is statistically significant, this study also conducts the multifactor ANOVA test to analyze the achieved results. For the ANOVA test, the computation time and the algorithm type are selected as the controlled factors and the average HVR or I_ε^1 of all the instances in one run is considered as the response variable. As MMBO has shown to be superior or has same performance over the MSA for all the tested instances, this study only tests OMABC, NSGA-II, IMABC and MMBO in the statistical analysis. Table 7 and 8 provide the detailed ANOVA results in terms of 1-HVR and I_ε^1; respectively. Here, if the value of P-value is less than 0.01, there is statistically statistical difference between the controlled factors or interactions. As you can see, ANOVA test shows that there exists a statistical difference between the tested algorithms, computation time and the interactions between them in terms of 1-HVR and I_ε^1. Meanwhile, the ANOVA test demonstrates that the proposed MBBO outperforms OMABC, NSGA-II and IMABC in terms of 1-HVR and I_ε^1 under all the three termination criteria.

Table 6 Results by all the implemented algorithms in terms of HVR and I_ε^1 when $\tau = 30$

Instance	HVR					I_ε^1				
	MSA	OMABC	NSGA-II	IMABC	MMBO	MSA	OMABC	NSGA-II	IMABC	MMBO
P7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
P8	1.000	1.000	1.000	0.996	1.000	1.003	1.001	1.001	1.030	1.000
P9	1.000	1.000	1.000	0.999	1.000	1.000	1.000	1.000	1.003	1.000
P11	1.000	0.995	0.995	0.992	0.998	1.000	1.038	1.044	1.060	1.017
P21	0.988	0.991	0.986	0.990	0.995	1.081	1.072	1.158	1.109	1.036
P25	0.983	0.987	0.988	0.985	0.990	1.091	1.077	1.088	1.092	1.060
P28	0.961	0.989	0.983	0.985	0.992	1.109	1.039	1.074	1.049	1.033
P29	0.927	0.966	0.962	0.962	0.978	1.231	1.170	1.237	1.237	1.116
P30	0.920	0.970	0.967	0.964	0.979	1.234	1.150	1.196	1.194	1.105
P32	0.921	0.987	0.984	0.990	0.990	1.180	1.051	1.067	1.043	1.043
P35	0.953	0.989	0.981	0.984	0.990	1.153	1.144	1.188	1.166	1.098
P45	0.952	0.987	0.979	0.984	0.989	1.170	1.052	1.118	1.079	1.045
P53	0.981	0.999	1.000	0.999	1.000	1.095	1.007	1.005	1.007	1.006
P58	0.877	0.972	0.947	0.961	0.976	1.283	1.128	1.321	1.244	1.090
P70	0.881	0.971	0.969	0.975	0.976	1.277	1.088	1.138	1.094	1.075
P75	0.882	0.966	0.963	0.968	0.971	1.251	1.106	1.195	1.154	1.100
P83	0.900	0.972	0.972	0.973	0.975	1.241	1.104	1.120	1.116	1.094
P89	0.948	0.967	0.964	0.976	0.965	1.273	1.260	1.392	1.275	1.217
P94	0.895	0.979	0.969	0.976	0.982	1.233	1.056	1.154	1.106	1.063
P111	0.881	0.971	0.967	0.972	0.974	1.323	1.073	1.129	1.105	1.073
P148	0.933	0.982	0.972	0.977	0.985	1.198	1.051	1.111	1.089	1.043
P297	0.832	0.963	0.972	0.971	0.971	1.338	1.109	1.123	1.133	1.085
Avg	0.939	0.982	0.978	0.981	**0.986**	1.168	1.081	1.130	1.108	**1.063**

*Best in bold

Table 7 ANOVA results in terms of 1-HVR

Source of variation	DF	SS	MS	F	P
Algorithm	3	0.0504	0.0168	5786.299	< 0.001
T	2	0.00352	0.00176	605.961	< 0.001
Algorithm $\times$ τ	6	0.000728	0.000121	41.792	< 0.001
Residual	108	0.000314	2.9E-06		
Total	119	0.055	0.000462		

Table 8 ANOVA results in terms of I_ε^1

Source of variation	DF	SS	MS	F	P
Algorithm	3	0.102	0.034	529.062	< 0.001
T	2	0.0263	0.0132	204.96	< 0.001
Algorithm $\times$ τ	6	0.00218	0.000364	5.668	< 0.001
Residual	108	0.00693	6.42E-05		
Total	119	0.137	0.00115		

Figure 7 illustrates the means plot of the interaction of the controlled factors in terms of 1-HVR and Fig. 8 illustrates the means plot of the interaction of the controlled factors in terms of I_ε^1 in the multifactor ANOVA test. Here, the overlapping internal means that there is no statistical insignificant difference between the values; there is statistically significant difference between the values when the internals are not overlapped. As seen from Fig. 7, the proposed MMBO is the best performer in terms of 1-HVR and it outperforms all the other algorithms statistically. From Fig. 8, it can be seen that the proposed MMBO is also the best performer in terms of I_ε^1 and it also

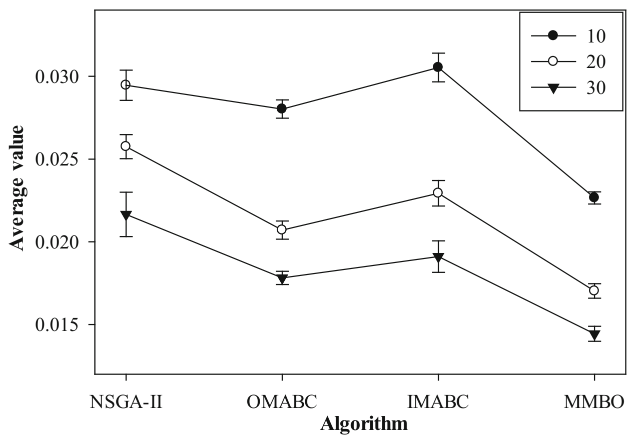

Fig. 7 Means plot and 95% Tukey HSD confidence intervals for the interactions between algorithms and computation time in terms of 1-HVR

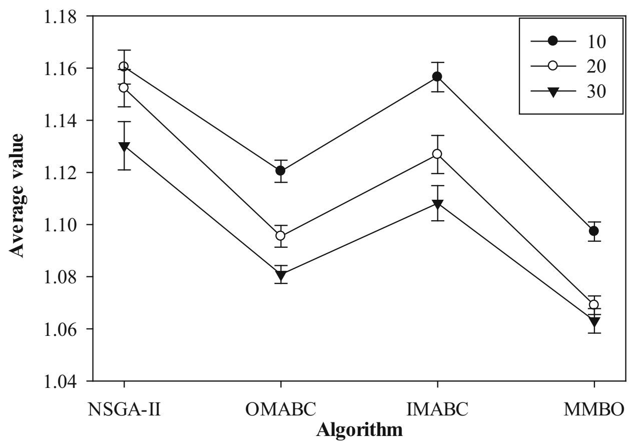

Fig. 8 Means plot and 95% Tukey HSD confidence intervals for the interactions between algorithms and computation time in terms of I_ε^1

outperforms all the other algorithms statistically. In summary, the statistical analysis validates the superiority of the proposed MMBO over the compared ones. In fact, the superiority of the MMBO is attributed to two main reasons as follows. 1) The leader improvement and block improvement help obtain the high-quality local optimal solutions in fast speed to emphasize exploitation. 2) The restart mechanism selects a solution from the permanent Pareto frontier or generate a solution randomly to replace the abandoned individual to emphasize exploration and help the algorithm to escape from being trapped into local optima.

Figure 9 illustrates the differences between empirical attainment functions (DEAFs) between MMBO and MSA (a), MMBO and OMABC (b), MMBO and NSGA-II (c) and MMBO and IMABC (d) when solving the P70 with 18 stations (ssss). All the algorithms solve this instance for 100 times to achieve the DEAF. Notice that MMBO outperforms the compared algorithm in one objective area when the value of this area is larger than 0.0; the compared algorithm outperforms the MMBO in one objective area when the value of this area is less than 0.0. From Fig. 9a, it is clear that MMBO outperforms MSA by achieving the better or the same performance in all the objective area. MMBO outperforms the OMABC, NSGA-II and IMABC slightly as seen in Fig. 9b, Fig. 9c and Fig. 9d by performing better in the larger objective area; respectively. In summary, this DEAF verified that the MMBO outperforms the MSA by a significant margin and MMBO outperforms the OMABC, NSGA-II and IMABC slightly.

6 Conclusions and future research

The collaborative robots (cobot) are increasingly explored for applications in the assembly line to embrace Industry 4.0, where cobots complete the tasks solely or help the human workers to complete the tasks. This study considers the multi-objective assembly line balancing problem with cobots to optimize the cycle time and the purchasing cost simultaneously, where different types of cobots are available and selected. To tackle this problem, this study formulates the multi-objective mixed-integer programming model, where this model can be solved utilizing the CPLEX solver with either objective or weighted objective. In addition, this study develops a multi-objective migrating bird optimization (MMBO) algorithm to achieve a set of high-quality Pareto solutions. The proposed MMBO algorithm proposes new population selection and update on the basis of the fast non-dominated sorting approach, and develops a restart mechanism to select one solution in the in the permanent Pareto archive to replace the abandoned solution which remains unchanged for several iterations.

To evaluate the performance of the developed method, this study conducts a comprehensive study and the proposed MMBO is compared with multi-objective non-dominated sorting genetic algorithm II, multi-objective simulated annealing algorithm and two multi-objective artificial bee colony algorithms. Computational study demonstrates that the multi-objective optimization is reasonable as the two objectives are conflicted in many occasions. And the proposed MMBO method obtains competing performance in comparison with the five reimplemented multi-objective algorithms. Due the promising performance of the developed MMBO, this multi-objective algorithm can be integrated into the decision support system to obtain a set of high-quality solutions for the line manager to select.

Since MMBO have several parameters that needs to be calibrated, further development of self-adaptive MMBO to adjust the parameters automatically is essential. There will

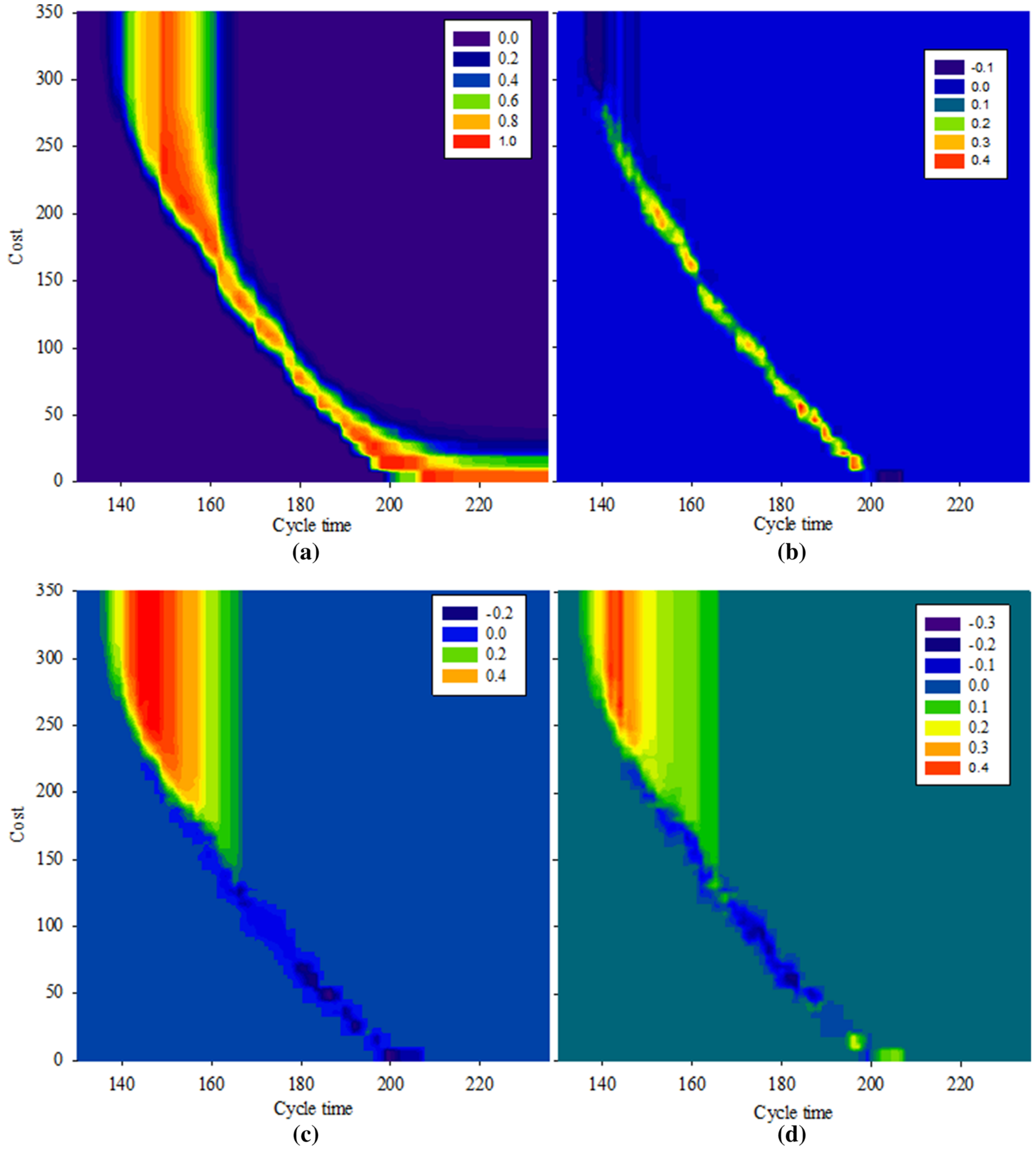

Fig. 9 DEAF between MMBO and MSA (**a**), MMBO and OMABC (**b**), MMBO and NSGA-II (**c**) and MMBO and IMABC (**d**)

be also a need to develop some simple and effective multi-objective algorithms. MMBO can also be enhanced by combing with the exact method, where exact method aims at solving the sub-problems optimally. Future study might also consider more realistic objectives such as maintenance costs of the cobots. Since cobots are bound to breakdown, it is necessary to study the uncertainty of the cobots or the rebalancing of the assembly line with cobots. It is also a good research opportunity to study the parallel working of a human worker and a cobot with multi-objective optimization as the parallel working is not considered in this study. It is also interesting to study how the utilization of the exact methods, e.g. branch, bound and remember algorithm [60] help in solving CRALBP optimally.

Acknowledgements This project is partially supported by National Natural Science Foundation of China under Grants 61803287 and 51875421 and China Postdoctoral Science Foundation under Grant 2018M642928.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solution approaches. Int J Prod Econ 142(2):259–277
2. Li Z, Janardhanan MN, Tang Q, Ponnambalam SG (2019) Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. Swarm Evolution Comput 50:100567. https://doi.org/10.1016/j.swevo.2019.100567
3. Weckenborg C, Kieckhäfer K, Müller C, Grunewald M, Spengler TS (2019) Balancing of assembly lines with collaborative robots. Bus Res. https://doi.org/10.1007/s40685-019-0101-y
4. Gupta S, Deep K (2019) A novel random walk grey wolf optimizer. Swarm Evol Comput 44:101–112. https://doi.org/10.1016/j.swevo.2018.01.001
5. Gupta S, Deep K, Heidari AA, Moayedi H, Chen H (2019) Harmonized salp chain-built optimization. Eng Comput. https://doi.org/10.1007/s00366-019-00871-5
6. Gupta S, Deep K (2019) Hybrid grey wolf optimizer with mutation operator. In: Bansal JC, Das KN, Nagar A, Deep K, Ojha AK (eds) Soft computing for problem solving 2019. Springer, Singapore, pp 961–968
7. Gupta S, Deep K, Engelbrecht AP (2020) A memory guided sine cosine algorithm for global optimization. Eng Appl Artif Intell 93:103718. https://doi.org/10.1016/j.engappai.2020.103718
8. Gupta S, Deep K (2020) A memory-based Grey Wolf Optimizer for global optimization tasks. Appl Soft Comput 93:106367. https://doi.org/10.1016/j.asoc.2020.106367
9. Gupta S, Deep K (2019) A hybrid self-adaptive sine cosine algorithm with opposition based learning. Expert Syst Appl 119:210–230. https://doi.org/10.1016/j.eswa.2018.10.050
10. Gupta S, Deep K (2018) Cauchy grey wolf optimiser for continuous optimisation problems. J Exp Theor Artif Intell 30(6):1051–1075. https://doi.org/10.1080/0952813X.2018.1513080
11. Li Z, Janardhanan MN, Ashour AS, Dey N (2019) Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem. Neural Comput Appl 31(12):9095–9111. https://doi.org/10.1007/s00521-018-3957-4
12. Janardhanan MN, Li Z, Bocewicz G, Banaszak Z, Nielsen P (2019) Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times. Appl Math Model 65:256–270. https://doi.org/10.1016/j.apm.2018.08.016
13. Eghtesadifard M, Khalifeh M, Khorram M (2020) A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017. Comput Ind Eng 139:106182. https://doi.org/10.1016/j.cie.2019.106182
14. Sewell EC, Jacobson SH (2012) A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS J Comput 24(3):433–442. https://doi.org/10.1287/ijoc.1110.0462
15. Li Z, Janardhanan MN, Rahman HF (2020) Enhanced beam search heuristic for U-shaped assembly line balancing problems. Eng Optim. https://doi.org/10.1080/0305215X.2020.1741569
16. Pape T (2015) Heuristics and lower bounds for the simple assembly line balancing problem type 1: overview, computational tests and improvements. Eur J Oper Res 240(1):32–42. https://doi.org/10.1016/j.ejor.2014.06.023
17. Huo J, Wang Z, Chan FTS, Lee CKM, Strandhagen JO (2018) Assembly line balancing based on beam ant colony optimisation. Math Probl Eng 2018:1–17. https://doi.org/10.1155/2018/2481435
18. Li Z, Kucukkoc I, Tang Q (2019) A comparative study of exact methods for the simple assembly line balancing problem. Soft Comput. https://doi.org/10.1007/s00500-019-04609-9
19. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713. https://doi.org/10.1109/TEVC.2008.919004
20. Li Z, Dey N, Ashour AS, Tang Q (2018) Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem. Neural Comput Appl 30(9):2685–2696. https://doi.org/10.1007/s00521-017-2855-5
21. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
22. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
23. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
24. Zhang Z, Tang Q, Li Z, Zhang L (2019) Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems. Int J Prod Res 57(17):5520–5537. https://doi.org/10.1080/00207543.2018.1530479
25. 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. J Clean Prod 215:744–756. https://doi.org/10.1016/j.jclepro.2019.01.030
26. Li Z, Tang Q, Zhang L (2016) Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm. J Cleaner Prod 135:508–522. https://doi.org/10.1016/j.jclepro.2016.06.131
27. Zhang Z, Tang Q, Ruiz R, Zhang L (2020) Ergonomic risk and cycle time minimization for the U-shaped worker assignment assembly line balancing problem: a multi-objective approach. Comput Oper Res 118:104905. https://doi.org/10.1016/j.cor.2020.104905
28. Rubinovitz J (1991) Design and balancing of robotic assembly lines. In: Proceedings of the fourth world conference on robotics research, Pittsburgh, PA, 1991
29. Rubinovitz J, Bukchin J, Lenz E (1993) RALB–a heuristic algorithm for design and balancing of robotic assembly lines. CIRP Ann 42(1):497–500. https://doi.org/10.1016/s0007-8506(07)62494-9
30. Borba L, Ritt M, Miralles C (2018) Exact and heuristic methods for solving the robotic assembly line balancing problem. Eur J Oper Res 270(1):146–156. https://doi.org/10.1016/j.ejor.2018.03.011
31. Kim H, Park S (1995) A strong cutting plane algorithm for the robotic assembly line balancing problem. Int J Prod Res 33(8):2311–2323. https://doi.org/10.1080/00207549508904817
32. Çil ZA, Mete S, Ağpak K (2017) Analysis of the type II robotic mixed-model assembly line balancing problem. Eng Optim 49(6):990–1009. https://doi.org/10.1080/0305215X.2016.1230208
33. Levitin G, Rubinovitz J, Shnits B (2006) A genetic algorithm for robotic assembly line balancing. Eur J Oper Res 168(3):811–825. https://doi.org/10.1016/j.ejor.2004.07.030
34. Gao J, Sun L, Wang L, Gen M (2009) An efficient approach for type II robotic assembly line balancing problems. Comput Ind Eng 56(3):1065–1080. https://doi.org/10.1016/j.cie.2008.09.027
35. Daoud S, Chehade H, Yalaoui F, Amodeo L (2014) Solving a robotic assembly line balancing problem using efficient hybrid methods. J Heuristics 20(3):235–259. https://doi.org/10.1007/s10732-014-9239-0
36. Nilakantan JM, Huang GQ, Ponnambalam S (2015) An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. J Clean Prod 90:311–325

37. Nilakantan M, Ponnambalam S (2016) Robotic U-shaped assembly line balancing using particle swarm optimization. Eng Optim 48(2):231–252
38. Nilakantan JM, Ponnambalam SG, Jawahar N, Kanagaraj G (2015) Bio-inspired search algorithms to solve robotic assembly line balancing problems. Neural Comput Appl 26(6):1379–1393. https://doi.org/10.1007/s00521-014-1811-x
39. Nilakantan JM, Nielsen I, Ponnambalam SG, Venkataramanaiah S (2017) Differential evolution algorithm for solving RALB problem using cost- and time-based models. Int J Adv Manuf Technol 89(1):311–332. https://doi.org/10.1007/s00170-016-9086-2
40. Li Z, Janardhanan MN, Nielsen P, Tang Q (2018) Mathematical models and simulated annealing algorithms for the robotic assembly line balancing problem. Assembly Autom 38(4):420–436. https://doi.org/10.1108/Aa-09-2017-115
41. Yoosefelahi A, Aminnayeri M, Mosadegh H, Ardakani HD (2012) Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. J Manuf Syst 31(2):139–151. https://doi.org/10.1016/j.jmsy.2011.10.002
42. Zhou B, Wu Q (2020) Decomposition-based bi-objective optimization for sustainable robotic assembly line balancing problems. J Manuf Syst 55:30–43. https://doi.org/10.1016/j.jmsy.2020.02.005
43. Pereira J, Ritt M, Vásquez ÓC (2018) A memetic algorithm for the cost-oriented robotic assembly line balancing problem. Comput Oper Res 99:249–261. https://doi.org/10.1016/j.cor.2018.07.001
44. Nilakantan JM, Li Z, Tang Q, Nielsen P (2017) Multi-objective co-operative co-evolutionary algorithm for minimizing carbon footprint and maximizing line efficiency in robotic assembly line systems. J Clean Prod 156:124–136. https://doi.org/10.1016/j.jclepro.2017.04.032
45. Zhou B, Wu Q (2019) An improved immune clonal selection algorithm for bi-objective robotic assemble line balancing problems considering time and space constraints. Eng Comput 36(6):1868–1892. https://doi.org/10.1108/ec-11-2018-0512
46. Li Z, Janardhanan MN, Tang Q, Nielsen P (2016) Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. Adv Mech Eng 8(9):1–14. https://doi.org/10.1177/1687814016667907
47. Rabbani M, Mousavi Z, Farrokhi-Asl H (2016) Multi-objective metaheuristics for solving a type II robotic mixed-model assembly line balancing problem. J Ind Prod Eng 33(7):472–484. https://doi.org/10.1080/21681015.2015.1126656
48. Aghajani M, Ghodsi R, Javadi B (2014) Balancing of robotic mixed-model two-sided assembly line with robot setup times. Int J Adv Manuf Tech 74(5–8):1005–1016. https://doi.org/10.1007/s00170-014-5945-x
49. Çil ZA, Mete S, Özceylan E, Ağpak K (2017) A beam search approach for solving type II robotic parallel assembly line balancing problem. Appl Soft Comput 61:129–138. https://doi.org/10.1016/j.asoc.2017.07.062
50. Li Z, Janardhanan MN, Tang Q, Nielsen P (2018) Mathematical model and metaheuristics for simultaneous balancing and sequencing of a robotic mixed-model assembly line. Eng Optim 50(5):877–893. https://doi.org/10.1080/0305215x.2017.1351963
51. Çil ZA, Mete S, Özceylan E (2018) A mathematical model for semi-robotic assembly line balancing problem: a case study. Int J Lean Think 9(1):70–76
52. Samouei P, Ashayeri J (2019) Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations. Appl Math Model 72:259–275. https://doi.org/10.1016/j.apm.2019.02.019
53. Dalle Mura M, Dini G (2019) Designing assembly lines with humans and collaborative robots: a genetic approach. CIRP Ann 68(1):1–4. https://doi.org/10.1016/j.cirp.2019.04.006
54. Yaphiar S, Nugraha C, Ma'ruf A (2020) Mixed model assembly line balancing for human-robot shared tasks. In: International manufacturing engineering conference & The Asia Pacific conference on manufacturing systems 2019, Singapore. iMEC-APCOMS 2019. Springer Singapore, pp 245–252
55. Rabbani M, Behbahan SZB, Farrokhi-Asl H (2020) The Collaboration of human-robot in mixed-model four-sided assembly line balancing problem. J Intell Robot Syst 100(1):71–81. https://doi.org/10.1007/s10846-020-01177-1
56. Zhang B, Pan Q-k, Gao L, Zhang X-l, Peng K-k (2019) A multi-objective migrating birds optimization algorithm for the hybrid flowshop rescheduling problem. Soft Comput 23(17):8101–8129. https://doi.org/10.1007/s00500-018-3447-8
57. Faccio M, Gamberi M, Bortolini M (2016) Hierarchical approach for paced mixed-model assembly line balancing and sequencing with jolly operators. Int J Prod Res 54(3):761–777
58. Ciavotta M, Minella G, Ruiz R (2013) Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. Eur J Oper Res 227(2):301–313. https://doi.org/10.1016/j.ejor.2012.12.031
59. Salehi M, Maleki HR, Niroomand S (2020) Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms. Neural Comput Appl 32:8217–8243. https://doi.org/10.1007/s00521-019-04293-8
60. Li Z, Çil ZA, Mete S, Kucukkoc I (2019) A fast branch, bound and remember algorithm for disassembly line balancing problem. Int J Prod Res. https://doi.org/10.1080/00207543.2019.1630774