Soft Computing (2023) 27:9639–9659
https://doi.org/10.1007/s00500-023-08130-y

OPTIMIZATION

Models and algorithms for U-shaped assembly line balancing problem with collaborative robots

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

Accepted: 29 March 2023 / Published online: 16 April 2023

Abstract
The collaborative robots (cobots) are increasingly being utilized in industries due to the advancement in the field of robotic technology and also due to the increase in labor costs. The cobots on the assembly line can be utilized to complete the tasks independently or assist the workers to complete the tasks. This study considers the U-shaped assembly line balancing problem with cobots, where several cobots with different purchasing costs are selected under the budget constraint. Three mixed-integer programming models are formulated to optimize the cycle time, and the built models are capable of solving the small-sized instances optimally. Two algorithms, artificial bee colony algorithm and migrating bird optimization algorithm, are developed and improved to tackle the large-sized instances, where new encoding scheme and decoding procedure are developed for this new problem. The computational tests demonstrate that the utilization of collaborative robots reduces the cycle time effectively in the assembly line. The comparative study on a set of instances shows that the proposed methodologies obtain competing performance in comparison with other 12 implemented algorithms.

Keywords Assembly line balancing · U-shaped assembly line · Human–robot collaboration · Collaborative robots · Metaheuristic

Abbreviations

ALBP	Assembly line balancing problem
RALBP	Robotic assembly line balancing problem
Cobot	Collaborative robots
CRALBP	Assembly line balancing problem with collaborative robots
UALBP	U-shaped assembly line balancing problem
RUALBP	Robotic U-shaped assembly line balancing problem
CRUALBP	U-shaped assembly line balancing problem with collaborative robots
MILP	Mixed-integer linear programming
ABC	Artificial bee colony algorithm
IABC	Improved artificial bee colony algorithm
MBO	Migrating bird optimization algorithm
IMBO	Improved migrating bird optimization algorithm

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

Zixiang Li
zixiangliwust@gmail.com

Qiuhua Tang
tangqiuhua@wust.edu.cn

Zikai Zhang
zhangzikai0703@gmail.com

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 lines have been widely utilized in modern industries to assemble products, and such systems help to improve the productivity in the production floor Battaïa and Dolgui (2013). The traditional assembly line consists of a set of connected stations and one worker is allocated to each station to perform the assembly tasks. On the assembly line, the assembly tasks are divided into several task sets and the workers on stations operate the corresponding task set from the first station to the last station in a sequence. The assignment of the tasks determines the line balance and based on this assembly line balancing problem

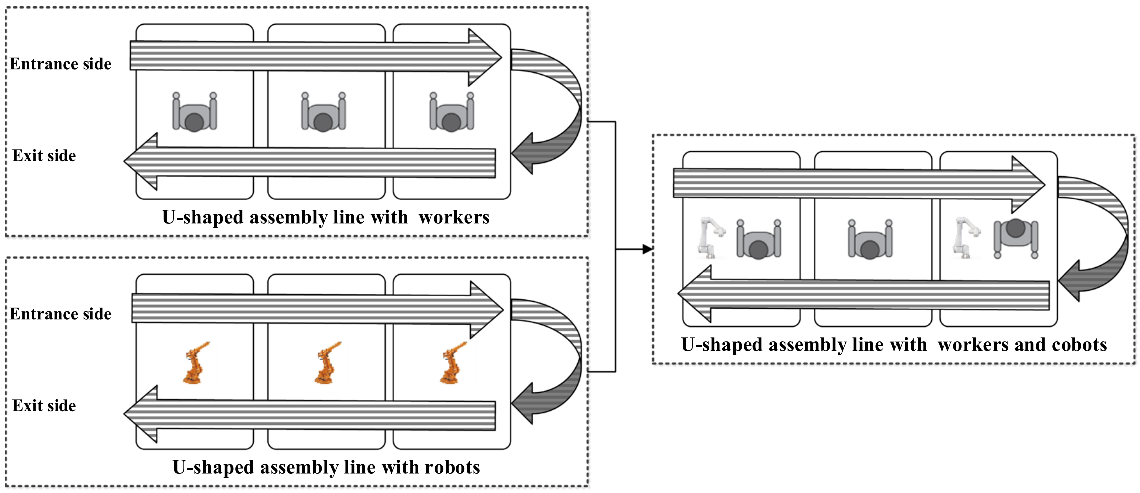
Entrance side
Exit side
U-shaped assembly line with workers
Entrance side
Exit side
U-shaped assembly line with robots
U-shaped assembly line with workers and cobots

Fig. 1 U-shaped assembly line with workers and cobots

(ALBP) is formulated to optimize the task assignment with one or several optimization criteria. There are two important constraints in the ALBP: precedence constraint and cycle time constraint. Precedence constraint demands that the predecessors of one task must be completed before operating this task; cycle time constraint demands that all the tasks on stations must be completed within the predetermined cycle time (CT).

U-shaped assembly line balancing problem (UALBP) has received increasing attentions since it offers higher flexibility than a normal assembly line. U-shaped assembly line could be divided into two sub-lines: the sub-line on the entrance side and the sub-line on the exit side. On the traditional U-shaped assembly line, human workers are allocated to the stations to operate all the tasks. The worker on one station first operates the tasks on the entrance side of this station, later moves to the exit side and operates the tasks on the exit side of this station, finally comes back to the entrance side. Faced with increased labor cost and concerns of efficiency, one of the trends that is being adopted in industries is the replacement of the human workers with robots. U-shaped assembly line with robots to complete all the tasks is referred to as robotic U-shaped assembly line (Li et al. 2019a). This type of line will help to reduced labor cost, higher accuracy and higher stability. The robotic U-shaped assembly line might be criticized for being expensive and lacks flexibility, another trend that is seen recently is to utilize collaborative robots, referred to as cobots, to operate some tasks solely or assist the human workers to operate some tasks (Weckenborg et al. 2020). The utilization of cobots achieves the human–robot collaboration, which combines the advantages of the human workers and cobots to help and improve the productivity of the system. Specifically, human–robot collaboration inherits the higher flexibility, higher adaptability and high decision-making skills of human workers and the high strength, endurance and accuracy of cobots (Weckenborg et al. 2020). When designing the U-shaped assembly line with cobots, U-shaped assembly line balancing problem with collaborative robots (CRUALBP) should be studied to optimize the task assignment and worker and robot allocation.

Figure 1 depicts the U-shaped assembly line with workers and cobots. In the U-shaped assembly line with workers, each station is allocated with one worker. The worker on one station first operates the tasks on the entrance side, later operates the tasks on the exit side and finally comes back to the entrance side to operate the corresponding tasks. In the U-shaped assembly line with robots, robots replace the workers completely to assemble the products and all the tasks are operated by robots. In the U-shaped assembly line with workers and cobots, one station might be allocated with one worker, one cobot or a pair of worker and cobot. On the station with one worker and one cobot, cobot can operate one task separately or assist the worker to operate the common tasks collaboratively.

Although there are some reported literature on robotic U-shaped assembly line balancing problem (RUALBP), there is no study tackling the CRUALBP. Hence, this study presents the first attempt to solve the considered CRUALBP, where different types of cobots with different cost should be selected and allocated to the stations within the maximum budget allowed. This study presents three main contributions as follows. 1) This study considers the CRUALBP where several cobots are selected within the budget constraint and this done for the first time in the literature. 2) Three mixed-integer linear programming (MILP) models are formulated to optimize the cycle time with the given station number. The developed MILP models can solve the small-sized instances optimally utilizing the CPLEX solver. 3) Improved artificial bee colony algorithm (IABC) and improved migrating bird optimization algorithm (IMBO) are developed to solve the large-sized instances within the acceptable computation time. The implemented algorithms utilize new encoding scheme and decoding procedure, and propose several improvements to achieve the proper balance between exploitation and exploration. The comprehensive comparative study is carried out to evaluate the performances of these algorithms, and the proposed algorithms obtain competing performance in comparison with other 12 implemented algorithms.

The structure of this paper is presented as follows. Section 2 provides the literature review and Sect. 3 presents the problem description and formulation. Subsequently, Sect. 4 describes the implemented metaheuristic methodologies and Sect. 5 illustrates a small-sized example. Section 6 carries out the comparative study, where the formulated model and algorithms are evaluated. Finally, Sect. 7 provides the conclusions and directions for future research.

2 Literature review

Since the first research paper published in 1955, several variants of ALBPs have been studied by many researchers (Battaïa and Dolgui 2013; Eghtesadifard et al. 2020). Based on the assembly layout, the assembly lines can be divided into straight line, U-shaped line, two-sided line, multi-manned line and parallel line. Based on the process alternative, there are three types: human worker, robot (or cobot) and worker and robot in collaboration. As there is no study considering all the characteristics of the problem in this study, this section mainly reviews the recent studies on the UALBP with human workers, RUALBP and assembly line balancing problem with collaborative robots (CRALBP).

In the basic UALBP with human workers, all the workers are considered to have the same skillset and only task assignment needs to be optimized. Miltenburg and Wijngaard (1994) presented the pioneering work on the UALBP and they develop the dynamic programming procedure to minimize the station number. Since then, many exact methods, heuristics and metaheuristics are developed. Regarding the exact methods, Scholl and Klein (1999) presented a branch and bound algorithm known as ULINO. Fattahi and Turkay (2015) formulated a corrected model to handle the precedence relation, and later Li et al. (2017) presented a new MILP model to solve the UALBP optimally. Li et al. (2018b) developed the branch, bound and remember algorithm to minimize the station number, which outperforms all the exact methods and is the best exact method. For the heuristic and metaheuristic methodologies, they include simulated annealing algorithm (Erel et al. 2001), genetic algorithm (Hwang et al. 2008), ant colony algorithms (Baykasoglu and Dereli 2009; Li et al. 2017; Sabuncuoglu et al. 2009), hybrid improvement heuristic approach (Özcan and Toklu 2009), critical path method (Avikal et al. 2013) and others, to cite just a few. Recently, Li et al. (2020) proposed an enhanced beam search algorithm, which outperforms the published algorithms and updates the upper bounds for many instance. Among these methodologies, branch, bound and remember algorithm is the best exact method (Li et al. 2018b) and the enhance beam search is state-of-the-art method for the UALBP (Li et al. 2020). There are also some studies considering the human workers with different skills, Oksuz et al. (2017) formulated the U-shaped assembly line worker assignment and balancing problem and they also develop the artificial bee colony algorithm and genetic algorithm to tackle it in short computational time. Afterward, Zhang et al. (2019a) presented an enhanced migrating birds optimization algorithm, which outperforms several other algorithms. Zhang et al. (2020) developed restarted iterated Pareto greedy algorithm to optimize the ergonomic risk and cycle time simultaneously, and this algorithm outperforms several other multi-objective algorithms in the comparative study.

For the studies on the RUALBP, Nilakantan and Ponnambalam (2016) formulated this problem to minimize the cycle time for the first time and develop the particle swarm optimization to solve this problem. Li et al. (2019a) presented two new MILP models with the objective of minimizing the cycle time and develop an improved migrating birds optimization algorithm, which outperforms seven other implemented algorithms. Zhang et al. (2019b) considered the energy consumption and the cycle time in the RUALBP. They build the multi-objective model and develop the Pareto artificial bee colony algorithm to achieve a set of non-dominated solutions. Zhang et al. (2019c) tackled the multi-objective RUALBP to minimize the carbon emission, noise emission and cycle time. They formulated this problem with a multi-objective model and developed the hybrid Pareto grey wolf optimization to solve this problem. The proposed algorithm outperforms five multi-objective algorithms in the comparative study.

In the literature, there are several different types of human–robot collaboration in the assembly lines and they are summarized as follows. Çil et al. (2018) formulated the semi-robotic ALBP and present a case study. Weckenborg and Spengler (2019) formulated the mathematical model to optimize the cost in the assembly line with collaborative robots taking into account ergonomics. Samouei and Ashayeri (2019) considered the semi-automated operations

Table 1 Summary of literature on the CRALBP

Literature (sorted by year)	Line type	Product type	Solution approaches
Çil et al. (2018)	Straight	Single	Model
Weckenborg and Spengler (2019)	Straight	Single	Model
Samouei and Ashayeri (2019)	Straight	Mixed	Two models
Dalle Mura and Dini (2019)	Straight	Single	Genetic algorithm
Weckenborg et al. (2020)	Straight	Single	Model and genetic algorithm
Yaphiar et al. (2020)	Straight	Mixed	Model
Çil et al. (2020)	Straight	Mixed	Model, bee algorithm and artificial bee colony algorithm
Rabbani et al. (2020)	Four-sided	Mixed	Model and particle swarm optimization algorithm
Li et al. (2021)	Straight	Single	Model and multi-objective migrating bird optimization algorithm
Koltai et al. (2021)	Straight	Single	Models
Nourmohammadi et al. (2022)	Straight	Single	Model and simulated annealing algorithm
Weckenborg et al. (2022)	Straight	Single	Model
This study	U-shaped	Single	Three models and two algorithms

in the mixed-model ALBP and they formulate two mathematical models to solve this problem. Dalle Mura and Dini (2019) presented a genetic algorithm to tackle the CRALBP. Weckenborg et al. (2020) formulates the CRALBP to minimize the cycle time, which is capable of solving the small-size instance optimally. They also develop a hybrid algorithm to tackle both the small-size and large-size instances. Yaphiar et al. (2020) presented a new MILP model for the mixed-model CRALBP. More recently, Çil et al. (2020) formulated the mixed-model ALBP with physical human–robot collaboration, which solves the small-size instance optimally. They also develop the improved bee algorithm and artificial bee colony algorithm to tackle the large-size instance efficiently. Rabbani et al. (2020) considered the human–robot collaboration in mixed-model four-sided ALBP. They formulated the MILP model and developed the particle swarm optimization algorithm to solve the considered problem. More recently, Li et al. (2021) formulated the model to minimize the cycle time and cobot cost. They also developed the Pareto migrating bird optimization algorithm to optimize two objectives simultaneously. Koltai et al. (2021) analyzed the task assignment and cycle times when robots are added with mathematical models. Nourmohammadi et al. (2022) optimized the cycle time and the number of cobots and workers in the CRALBP, and they provided the mathematicl model and simulated annealing algorithm to solve the CRALBP. Weckenborg et al. (2022) considered the ergonomic and economic objectives with collaborative robots and exoskeletons utilizing one mixed-integer programming model. The literature on the CRALBP are summarized in Table 1.

From the literature review presented, the problem setting is different from that in the related and published studies and there is no study considering the CRUALBP. To fill this gap, this study formulates this problem for the first time and proposes two recent algorithms to solve this problem in reasonable computation time.

3 Problem description and formulation

This section first provides the problem description in Sect. 3.1 and later presents the MILP model of this problem in Sect. 3.2.

3.1 Problem description

For the human–robot collaboration, there are five scenarios in the assembly lines (Hashemi-Petroodi et al. 2020; Krüger et al. 2009) as follows. And all of them have real applications in the diverse industrial context. 1) Shared workspace, separate cooperation and sequential operations: A worker and a cobot are allocated to the same station but not the common workspace (e.g., the left side and the right side of one station), and they operate the tasks separately and they can only operate the tasks in sequence. 2) Shared workspace, separate cooperation and simultaneous operations: A worker and a cobot are allocated to the same station but not the common workspace, and they operate the tasks separately and they can operate the different tasks in parallel simultaneously. 3) Shared workspace, collaborative cooperation and sequential operations: A worker and a cobot are allocated to the same station but not the common workspace, and they can perform a common task in

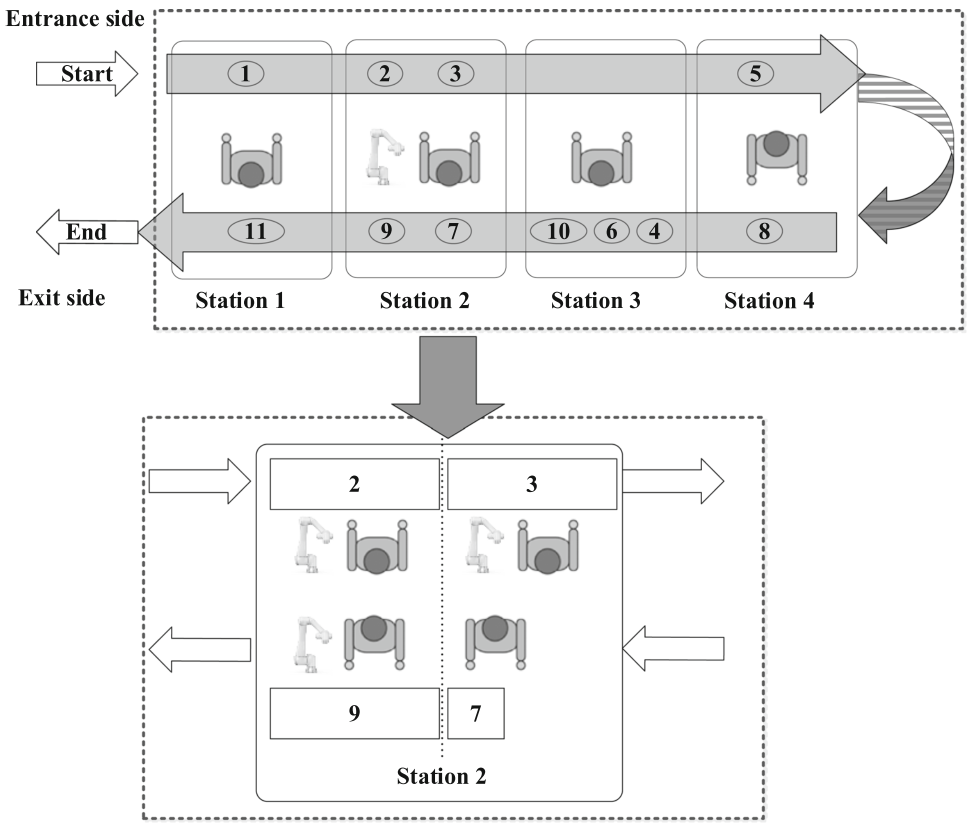

Fig. 2 Task assignment and worker and cobot allocation on U-shaped assembly line

collaboration and they can only operate the tasks in sequence. 4) Shared workspace, collaborative cooperation and simultaneous operations: A worker and a cobot are allocated to the same station but not the common workspace, and they can perform a common task in collaboration and they can operate the different tasks in parallel simultaneously. 5) Common workspace, collaborative cooperation and sequential operations: A worker and a cobot are allocated to the same station and the same workspace, and they can perform a common task in collaboration and they can only operate the tasks in sequence.

For the considered CRUALBP problem, a worker and a cobot can be allocated to the same station and they can complete the tasks solely or in collaboration, but they cannot operate the different tasks in parallel simultaneously. Namely, this study takes the first scenario, the third scenario and the fifth scenario into account. The applications of second scenario and the fourth scenario might refer to Weckenborg et al. (2020). In the CRUALBP, one station might be allocated with a worker, a cobot and a pair of worker and cobot. If one worker and robot are allocated to the same station, one task on this station is operated by one of the three process alternatives: operated by a human worker, operated by cobot and operated by the worker and cobot in collaboration. Hence, the CRUALBP consists of three interrelated sub-problems: 1) task assignment to station; 3) the allocation of the workers and cobots; 3) the selection of the process alternative. And there are three main constraints needed to be considered when solving the CRUALBP: precedence constraint, cycle time constraint and budget constraint. Precedence constraint demands that one task can be assigned when all its predecessors or successors have been assigned. Cycle time constraint demands that all the task must be completed within the cycle time. Budget constraint requires that the costs of buying cobots must be equal to or less than the maximum budget provided in advance.

Figure 2 provides an example task assignment and worker and cobot allocation on U-shaped assembly line. As you can see, there are four stations utilized and 11 tasks are allocated to the entrance side and the exit side of this line. And one worker, one worker and one cobot, one worker and one worker are allocated to station 1, station 2, station 3 and station 4, respectively. For the task assignment on station 2, it is observed that task 2 is allocated to the entrance side and is operated by the worker and cobot in collaboration, task 3 is allocated to the entrance side and is operated by the worker and cobot in collaboration, task 7 is allocated to the exit side and is operated by the worker and finally task 9 is allocated to the exit side and is operated by the worker and cobot in collaboration.

The main assumptions of the considered CRUALBP are provided at first based on Weckenborg et al. (2020).

(1) One type of product is produced on the U-shaped assembly line.
(2) The precedence relations between tasks and the operation times of tasks by process alternatives are known and fixed.
(3) The operation times of tasks are determined by the process alternative and one task might need different time to be completed when being operated by different process alternatives
(4) Only one type of worker is considered (the workers are assumed to have the same skills).
(5) There are different types of cobots available and they might have different skills and different purchasing cost.
(6) The total cost of cobots must be within the maximum budget allowed to buy the cobots.
(7) It is not considered that a worker and cobot operate different tasks at the same time on the same station (the second scenario and the fourth scenario are not considered).
(8) The setup times, loading and unloading time and maintenance operation are not considered.

3.2 Model formulation

Based on Li et al. (2017), this section formulates three mathematical models: Model 1, Model 2 and Model 3 for the considered problem. For clarification, the indices, parameters and decision variables in these models are first explained as follows. Among the indices and parameters, the p is utilized to indicate the process alternative. For one task, there are $2 \cdot nr + 1$ process alternatives available. Here, this study sets that $p = 1$ when one human worker operates one task, $p = 1 + r$ when cobot r operates one task solely, and $p = \mathrm{nr} + 1 + r$ when a worker and cobot r operate one task collaboratively.

Indices and parameters:

nt	Number of tasks.
ns	Number of available stations.
nr	Types of available cobots.
i, j	Task index; $i,j \in \{1, 2, \ldots, nt\}$.
k, l	Station index; $k, l \in \{1, 2, \ldots, ns\}$.
r	Cobot index; $r \in \{1, 2, \ldots, nr\}$.
p	Process alternative index, $p \in \{1, 2, \ldots, nr + 1, nr + 2, \ldots, 2 \cdot nr + 1\}$.
I	Set of tasks, $I = \{1, 2, \ldots, nt\}$.
K	Set of stations, $K = \{1, 2, \ldots, ns\}$.
R	Set of cobot types, $R = \{1, 2, \ldots, nr\}$.
P	Set of process alternatives, $P = \{1, 2, \ldots, nr + 1, nr + 2, \ldots, 2 \cdot nr + 1\}$.
t_{ip}	Operation time of task i by processing alternative p.
$\wp$	Set of pairs of tasks ($\wp = \{\cdots, r, \cdots\}$), where $r = (i,j)$ is one pair of tasks in which task i is the immediate predecessor of task j.
c_r	Purchasing cost of cobot r.
ψ	A very large positive number.
MB	Maximum budget allowed to buy the cobots.
MW	Maximum number of available workers.

Decision variables:

CT	Cycle time.
a_{ikp}	1, if task i is operated by process alternative p on station k; 0, otherwise. (Only utilized in Model 1)
u_i	1, if task i is allocated to the entrance side of one station; 0, otherwise. (Only utilized in Model 1)
x_{ikp}	1, if task i is operated by process alternative p on the entrance side of station k; 0, otherwise. (Only utilized in Model 2 and Model 3)
y_{ikp}	1, if task i is operated by process alternative p on the exist side of station k; 0, otherwise. (Only utilized in Model 2 and Model 3)
w_{rk}	1, if cobot r is allocated to station k; 0, otherwise.
v_k	1, if a human worker is assigned to station k; 0, otherwise.

The main difference between the CRALBP and CRUALBP is the precedence constraint. On the straight line, one task is assignable when all its predecessors have been allocated; on the U-shaped line, one task is assignable to the entrance side when all its predecessors have been allocated or the exist side when all its successors have been allocated. On the basis of the model in Fattahi and Turkay (2015), the first mathematical model, referred to as Model 1, is formulated utilizing expression (1) to expression(11), where the decision variable a_{ikp} is utilized to determine the allocated station that one task is allocated to and decision variable u_i is proposed to determine the side (entrance side or exit side) that one task is allocated to.

Expression (1) is objective function to minimize the cycle time. Constraint (2) is the task occurrence constraint and it demands that each task must be allocated to one station and operated by one process alternative available. Constraint (3) and constraint (4) deal with the precedence constraint. Constraint (3) indicates that the predecessor i of task j must be allocated to the former or same station when both of them are allocated to the entrance side; Constraint (4) indicates that the predecessor i of task j must be allocated to the latter or same station when both of them are allocated to the exit side. Constraint (5) deals with the cycle time constraint and it demands that the total operation time on one station must be less than or equal to the cycle time. Constraint (6) and constraint (7) determine the cobot allocation. Constraint (6) requires that cobot r must be allocated to station k when there is one task on this station is operated by cobot r solely and a worker and cobot r in collaboration. Constraint (7) demands that at most one cobot is allocated to one station and it is not allowed for one station to have more than one cobot. Constraint (8) is the budget constraint, indicating the total cost of buying the cobots must be less than or equal to the maximum budget. Constraint (9) determines the worker allocation and it requires that a worker must be allocated to station k when there is one task on this station is operated by a worker solely and a worker and one cobot in collaboration. Constraint (10) restricts that the number of workers is less than or equal to the maximum number of available workers. Constraint (11) restricts that a_{ikp}, u_i, w_{rk} and v_k are the 0–1 variables.

$$\text{Min CT} \tag{1}$$

$$\sum_{k\in K}\sum_{p\in P} a_{ikp} = 1 \quad \forall i \in I \tag{2}$$

$$\sum_{k\in K}\sum_{p\in P} k\cdot a_{ikp} - \sum_{l\in K}\sum_{p\in P} l\cdot a_{jlp} \le ns\cdot\left(1+u_i-2u_j\right) \quad \forall (i,j) \in \wp \tag{3}$$

$$\sum_{k\in K}\sum_{p\in P} k\cdot a_{jkp} - \sum_{l\in K}\sum_{p\in P} l\cdot a_{ilp} \le ns\cdot u_i \quad \forall (i,j) \in \wp \tag{4}$$

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

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

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

$$\sum_{r\in R}\left(c_r\cdot\sum_{k\in K} w_{\mathrm{rk}}\right) \le MB \tag{8}$$

$$\psi\cdot v_k \ge \sum_{i\in I} a_{ik,1} + \sum_{i\in I}\sum_{p\in\{nr+2,\cdots,2\cdot nr+1\}} a_{ikp} \; \forall k \in K \tag{9}$$

$$\sum_{k\in K} v_k \le \mathrm{MW} \tag{10}$$

$$a_{ikp}, u_i, w_{rk}, v_k \in \{0,1\} \tag{11}$$

The second model, referred to as Model 2, is built based on Urban and Chiang (2006). Model 2 utilizes the x_{ikp} and y_{ikp} to determine the task assignment and is formulated with expression (12) to expression (22). The objective in expression (12) optimizes the cycle time. Constraint (13) is the occurrence constraint, indicating that one task must be allocated to the entrance side or exit side of one station and be operated by one process alternative available. Constraint (14) and constraint (15) tackle the precedence constraint. Constraint (14) demands that the predecessor i of task j must be allocated to the former or same station when both of them are allocated to the entrance side; Constraint (15) demands that the predecessor i of task j must be allocated to the latter or same station when both of them are allocated to the exit side. Constraint (16) is the cycle time constraint. Constraint (17) and constraint (18) deal with cobot allocation and constraint (19) is the budget constraint. Constraint (20) deals with the worker assignment and constraint (21) restricts the number of human workers.

$$\text{Min } CT \tag{12}$$

$$\sum_{k\in K}\sum_{p\in P}\left(x_{ikp}+y_{ikp}\right) = 1 \quad \forall i \in I \tag{13}$$

$$\sum_{k\in K}\sum_{p\in P}(\mathrm{ns}-k+1)\cdot\left(x_{ikp}-x_{jkp}\right) \ge 0 \;\; \forall (i,j) \in \wp \tag{14}$$

$$\sum_{k\in K}\sum_{p\in P}(\mathrm{ns}-k+1)\cdot\left(y_{jkp}-y_{ikp}\right) \ge 0 \;\; \forall (i,j) \in \wp \tag{15}$$

$$\sum_{i\in I}\sum_{p\in P} t_{ip}\cdot\left(x_{ikp}+y_{ikp}\right) \le CT \quad \forall k \in K \tag{16}$$

$$\psi\cdot w_{\mathrm{rk}} \ge \sum_{i\in I}\left(x_{ik,1+r} + y_{ik,1+r} + x_{ik,nr+1+r} + y_{ik,nr+1+r}\right) \; \forall k \in K, r \in R \tag{17}$$

$$\sum_{r\in R} w_{\mathrm{rk}} \le 1 \; \forall k \in K \tag{18}$$

$$\sum_{r\in R}\left(c_{\mathrm{r}}\cdot\sum_{k\in K} w_{\mathrm{rk}}\right) \le MB \tag{19}$$

$$\psi \cdot v_k \geq \sum_{i\in I}(x_{ik,1} + y_{ik,1}) + \sum_{i\in I}\sum_{p\in\{nr+2,\cdots,2\cdot nr+1\}}(x_{ikp} + y_{ikp})\ \forall k \in K \tag{20}$$

$$\sum_{k\in K} v_k \leq \mathrm{MW} \tag{21}$$

$$x_{ikp}, y_{ikp}, w_{\mathrm{rk}}, v_{\mathrm{k}} \in \{0, 1\} \tag{22}$$

The third model, referred to as Model 3, is formulated based on Model 2 and Li et al. (2017). Model 3 has the same decision variables as Model 2, whereas Model 3 utilizes constraint (23) to replace constraint (14) and constraint (15). Namely Model 3 consist of expression (12), constraint (13), constraint (23) and constraints (16–22). Here, constraint (23) is the precedence constraint and it demands that, for a pair of task i of task j where task i is the predecessor of task j, 1) task i must be allocated to the former or same station when both of them are alloated to the entrance side, 2) task i must be allocated to the latter or same station when both of them are allocated to the exit side, 3) task i must be allocated to the entrance side when they are allocated to entrance side and exit side, respectively.

$$\sum_{k\in K}\sum_{p\in P} k\cdot(x_{ikp} - x_{jkp}) + \sum_{k\in K}\sum_{p\in P}(2\cdot ns - k)\cdot(y_{ikp} - y_{jkp}) \leq 0\ \forall (i,j) \in \wp \tag{23}$$

All the three models are MILP models and they are capable of tackling the small-size instances optimally utilizing the CPLEX solver. This study will further evaluate and compare these models in Sect. 6.2.

4 Implemented metaheuristic methodologies

As the considered problem is one NP-hard problem in nature, the formulated models cannot achieve satisfying solutions when solving large-size instance. Hence, this section develops two recent algorithms to tackle the large-size instances effectively. These algorithms utilize new encoding scheme and decoding procedure to achieve feasible solutions. The encoding and decoding, applied algorithms and the utilized neighbor structures are presented in the following subsections in detail.

4.1 Encoding and decoding

As the CRUALBP involves task assignment and the worker and cobot allocation, this study utilizes two vectors for encoding: permutation vector and process alternative selection vector. Task permutation vector determines the task assignment; process alternative selection vector determine the worker and cobot allocation. Figure 3 illustrates one example solution presentation with 11 tasks, four stations and four types of cobots. In the decoding scheme, task permutation is one sequence of all the tasks and the tasks in the former positions are supposed to have the higher priorities. For instance, task 1 and task 11 are in the first two positions and they are allocated to the first station in the solution presentation. Process alternative selection vector is one set of process alternatives on stations. If the code in the k th position is p, it donates that process alternative p should be allocated to station k. For instance, one worker with the p of 1 is allocated to the first station and one worker and cobot 3 with the p of 8 are allocated to the second station.

As the encoding scheme is not a feasible solution, this study develops the one decoding procedure in Algorithm 1, where the CT_{Init} is the initial cycle time. As you can see, the worker and cobot allocation is first determined utilizing the process alternative selection vector. Subsequently, the tasks are assigned to the entrance side and exit side of stations in sequence. During task assignment, one task is assignable when it satisfies the precedence constraint and the cycle time constraint. Precedence constraint demands that the predecessors or successors have been assigned; cycle time constraint demands that this task can be completed within the initial cycle time CT_{Init} by at least one process alternative when $k < ns$. It is to be noted that, this decoding procedure ignores the cycle time for the last station to obtain the feasible solution for each solution presentation.

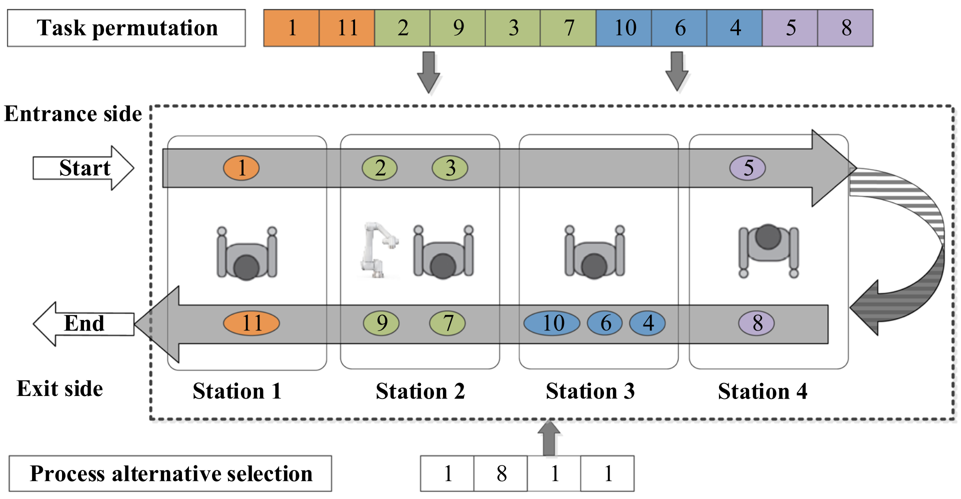

Fig. 3 Solution presentation

```
Algorithm 1: Decoding procedure for the CRUALBP
% Determine the worker and cobot allocation
Determine the worker and cobot allocation directly utilizing the process alternative selection vector;
% Determine the task assignment
For $k := 1$ to $ns$
  While (1)
    Achieve the assignable task set by checking the precedence constraint and cycle time constraint;
    % One task is assignable when 1) its predecessors or successors have been allocated; 2) this task can be completed within the initial cycle time $CT_{Init}$ by at least one process alternative when $k < ns$.
    Terminate the while loop when the assignable task set is empty; otherwise, execute the following steps.
    Select one assignable task which is in the former position of the task permutation;
    Select the process alternative with the smallest operation time as the process alternative for the selected task;
    Update the remained capacity of the current station;
  Endwhile
Endfor
```

How to determine the proper initial cycle time CT_{Init} is one important factor when solving the type II CRUALBP. One possible method is determining the best cycle time with increments. This method starts testing CT_{LB} as the initial cycle time CT_{Init}, where $CT_{LB} = \left(\sum_i \min_p t_{ip}\right)/\mathrm{ns}$.

Subsequently, CT_{Init} is updated with $\mathrm{CT_{Init}} \leftarrow \mathrm{CT_{Init}} + 1$ until the tasks of the last station can be completed within CT_{Init}. One possible drawback of this method is that the decoding procedure is conducted for many times to obtain the proper initial cycle time for one encoding scheme. Hence, on the basis of Li et al. (2019a), Li et al. (2019b) and several others, this study proposes the iterative mechanism to update the initial cycle time during the algorithm' evolution. The proposed iterative mechanism is presented in Algorithm 2, where CT_{Best} is the best cycle time during the evolution. It is clear that all the solutions utilize the same initial cycle time and the cycle time is reduced gradually utilizing the $\mathrm{CT_{Init}} \leftarrow \mathrm{CT_{Best}} - 1$. The preliminary experiments show that iterative mechanism produces superior performance and hence it is utilized in this study.

Algorithm 2: Iterative mechanism to update the initial cycle time

```
Step 1: Set a large positive number as the initial cycle time $CT_{Init}$, where
$CT_{Init} = 2 \cdot \left(\sum_i \min_p t_{ip}\right)/ns$;
Step 2: Decode with the initial cycle time $CT_{Init}$;
Step 3: Update $CT_{Best}$ with the better cycle time and update $CT_{Init}$ with
$CT_{Init} \leftarrow CT_{Best} - 1$ when new best cycle time is achieved, and re-decode
utilizing the new initial cycle time $CT_{Init}$;
Step 4: Terminate the algorithm when the termination criterion is met;
otherwise, execute Step 2;
```

4.2 Improved artificial bee colony algorithm

The proposed IABC is adopted from that in (Çil et al. 2020), where IABC is the best performer among the 11 compared algorithms. The main procedure of the IABC is provided in Algorithm 3. Here, parameter PS is the population size, parameter *limit* is the number of times before abandoning the individual remained unchanged, and parameter r within $[0, 1]$ is the probability to conduct the local search. IABC starts with initializing the swarm, and afterward conducts the modified employed bee phase, modified onlooker phase, modified scout phase and local search until meeting the termination criterion.

IABC has made following modifications on the employed bee phase, onlooker phase and scout phase, and also optionally utilizes the local search. In this algorithm, modified onlooker phase selects the best and non-duplicated PS solutions from the incumbent swam and the new swam to obtain a high-quality and diverse swarm. Modified scout phase replaces the abandoned individuals with the high-quality neighbor solution. Meanwhile, local search aims at emphasizing the exploitation capacity. In short, the modifications enhance the exploitation and exploration capacity of the proposed IABC method. As you will see in Sect. 6.3, the proposed IABC outperforms the original artificial bee colony algorithm by a significant margin.

Algorithm 3: Procedure of the proposed IABC

```
Initialize PS individuals;
do
    For p:=1 to PS        % Modified employed bee phase
      $p'$ ← Neighbor solution of individual $p$;
      $p \leftarrow p'$ when $\mathrm{Fit}(p') \leq \mathrm{Fit}(p)$;
    Endfor
    For $p$:=1 to PS        % Modified onlooker phase
      Select one individual $s$ utilizing binary tournament selection;
      $s'$ ←Neighbor solution of individual s;
    Endfor
    Select the best and non-duplicated PS solutions (including all the $s$ and $s'$);
    % Modified scout phase
    For one individual $p$ when it is a duplicated individual or remained unchanged for limit times
      Obtain a set of neighbor solutions of individual $p$ by conducting the neighbor operators for several times;
      Select the best neighbor solution $p'$ (different from the $p$) and update $p \leftarrow p'$;
    Endfor
    For one individual $p$ when $\mathrm{Rand}[0,1] \leq r$  % Local search
    % $\mathrm{Rand}[0,1]$ is a random number within$[0,1]$
      For $k$:=1 to $nt$ do
        $p'$ ← Neighbor solution of individual $p$;
        $p \leftarrow p'$ when $\mathrm{Fit}(p') \leq \mathrm{Fit}(p)$;
      Endfor
    Endfor
Until (Termination criterion is met)
```

4.3 Improved migrating bird optimization algorithm

The proposed IMBO is adopted from (Janardhanan et al. 2019), where IMBO outperforms five other algorithms. The main procedure of the IMBO is illustrated in Algorithm 4. The proposed IMBO include four original parameters and three new parameters. The four original parameters include the population size (PS), the consecutive times before the leader replacement (m), the number of neighbor solutions for the leader (k) and the number of shared neighbors (x). The three new parameters include the initial temperature (T_{init}), the cooling rate (α) and parameter *rt* where T is re-initialized with $T = T_{\mathrm{init}}$ when the best cycle time has not been updated for *rt* times. The proposed IMBO starts with initializing the population, and afterward conducts the modified leader improvement, modified block improvement and leader replacement repeatedly until the termination criterion is met. In the main loop, leader replacement is conducted after executing the modified leader improvement and modified block improvement for m consecutive times.

The modified leader improvement and modified block improvement propose several modifications to enhance the exploitation and exploration capacity as follows. 1) The incumbent individual is replaced with the neighbor solution once the neighbor solution achieves the same or better performance. This modification speeds up the evolution process by replacing the poor-quality individual. 2) New acceptance criterion based on the simulated annealing algorithm is developed to accept the worse solution with a certain probability. Especially, if the best cycle time has not been updated for *rt* times, the temperature is re-initialized to increase the possibility of accepting the worse solution. This utilization of new acceptance helps the algorithm to escape from being trapped into local optima. 3) The neighbor solution of one individual is set to a very large number when it shares the same fitness value of the original individual. In other words, the neighbor solutions with the same fitness value of the original individual cannot be shared by the individual in the back. This modification aims at preventing the premature of the algorithm by preserving the diversity of the population. As you will see in Sect. 6.3, the proposed IMBO outperforms the original migrating bird optimization algorithm statistically.

4.4 Utilized neighbor structure

The neighbor structures have important effect on the algorithms' performance. As this encoding scheme has two vectors, it is necessary to design effective neighbor operators to modify the two vectors to obtain high-quality neighbor solutions. As seen in Fig. 4, this study utilizes the swap operator and insert operator to modify the task permutation vector. Swap operator selects two tasks in different positions randomly and exchange the positions of the two selected tasks. Inset operator selects one task randomly and insert it in a new position different from its original position. As seen in Fig. 4, to modify the process

Algorithm 4: Procedure of the proposed IMBO

```
Initialize PS individuals;
Set $T \leftarrow T_{init}$;
do
    For i:=1 to m do
        % Modified leader improvement
        For j:=1 to k do
          $s' \leftarrow$ Neighbor solution of leader s;
          $s \leftarrow s'$ when $\mathrm{Fit}(s') \le \mathrm{Fit}(s)$;
        Endfor
        $s'' \leftarrow$ Best one of the k neighbor solutions;
        $s \leftarrow s''$ when $\mathrm{Fit}(s'') \le \mathrm{Fit}(s)$ or $\mathrm{Rand}[0,1] \le exp^{-(\mathrm{Fit}(s'')-\mathrm{Fit}(s))/(T\times Fit(s))}$;
        Set $\mathrm{Fit}(s')$ as a very large number when $\mathrm{Fit}(s') = \mathrm{Fit}(s)$ for each solution $s'$;
        % Modified block improvement
        For one individual on left and right sides
          For j:=1 to (k-x) do
          $p' \leftarrow$ Neighbor solution of individual p;
          $p \leftarrow p'$ when $\mathrm{Fit}(p') \le \mathrm{Fit}(p)$;
          Endfor
          $p'' \leftarrow$ Best one of the (k-x) neighbor solutions $p'$ and x unused best neighbor solution of
          the individual in the front;
          $p \leftarrow p''$ when $\mathrm{Fit}(p'') \le \mathrm{Fit}(p)$ or $\mathrm{Rand}[0,1] \le exp^{-(\mathrm{Fit}(p'')-\mathrm{Fit}(p))/(T\times Fit(p))}$;
          Set $\mathrm{Fit}(p')$ as a very large number when $\mathrm{Fit}(p') = \mathrm{Fit}(p)$ for each solution $s'$;
        Endfor
        Update $T \leftarrow T \times \alpha$;
        Update $T \leftarrow T_{init}$ when the best cycle time has not been updated for rt times;
    Endfor
    % Leader replacement
    Select one side randomly and move the leader to the end of the selected side;
    Forward the following individuals of the selected side;
Until (Termination criterion is met)
```

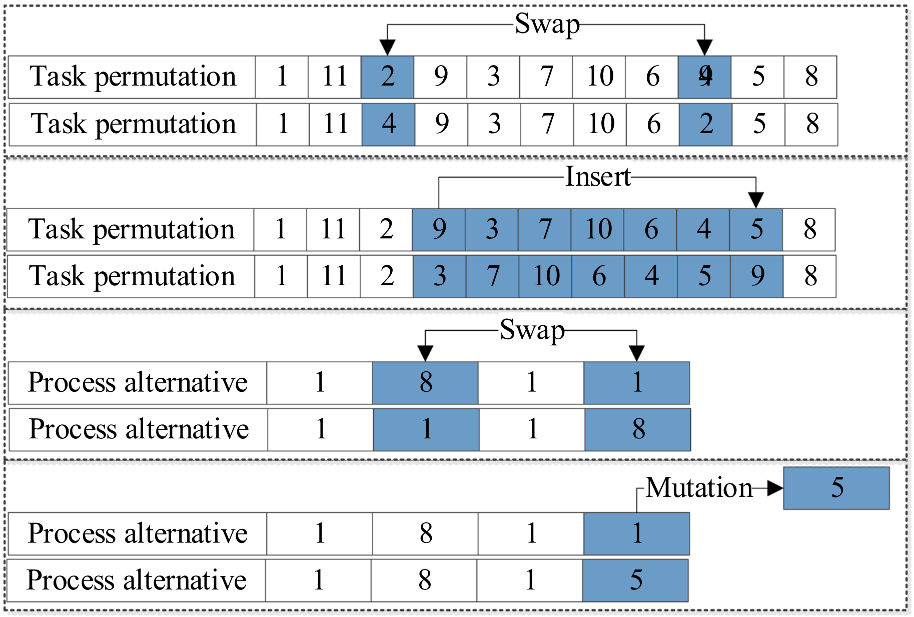
Swap
Task permutation 1 11 2 9 3 7 10 6 4 5 8
Task permutation 1 11 4 9 3 7 10 6 2 5 8
Insert
Task permutation 1 11 2 9 3 7 10 6 4 5 8
Task permutation 1 11 2 3 7 10 6 4 5 9 8
Swap
Process alternative 1 8 1 1
Process alternative 1 1 1 8
Mutation
5
Process alternative 1 8 1 1
Process alternative 1 8 1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

Fig. 4 Utilized neighbor operators

Table 2 Precedence relation between tasks and the operation times of tasks

Task	Successors	Operation times of tasks 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

Table 3 Achieved cycle times under different budgets

Straight assembly line		U-shaped assembly line	
Maximum budget	Cycle time	Maximum budgets	Cycle time
0.0	12	0.0	12
10.0	12	10.0	12
20.0	11	20.0	10
30.0	10	30.0	10
40.0	10	40.0	9
50.0	9	50.0	9
60.0	9	60.0	9
70.0	9	70.0	8
80.0	9	80.0	8

alternative selection vector, this study utilizes both swap operator and mutation operator. Swap operator selects two process alternatives in two different stations randomly and exchanges the selected two process alternatives. Mutation operator selects one process alternative on one station randomly and replaces it with a different process alternative. As two vectors are involved and each vector has two neighbor operators, for one individual, this study first selects one vector randomly to be modified and later selects one of the two corresponding neighbor operators to modify the selected vector.

5 An illustrated example

This section provides an example to highlight the features of the considered problem. This instance has 11 tasks, four types of cobots ($2 \cdot 4 + 1$ process alternatives) and four stations. Table 2 illustrates the precedence relation and the operation times of task by process alternatives ($P = \{1, 2, \cdots, 4+1, 4+2, \cdots, 2 \cdot 4 + 1\}$). The cost of these four cobots are 10.11, 12.79, 18.55 and 20.83 units.

Table 3 presents the achieved cycle times under different budgets on the straight assembly line and U-shaped assembly line. As you can see, the cycle time could be reduced when the budget increase. Specifically, on the U-shaped assembly line, the cycle time is 12 when the budget is 0.0 (no cobot is utilized). And the cycle time is

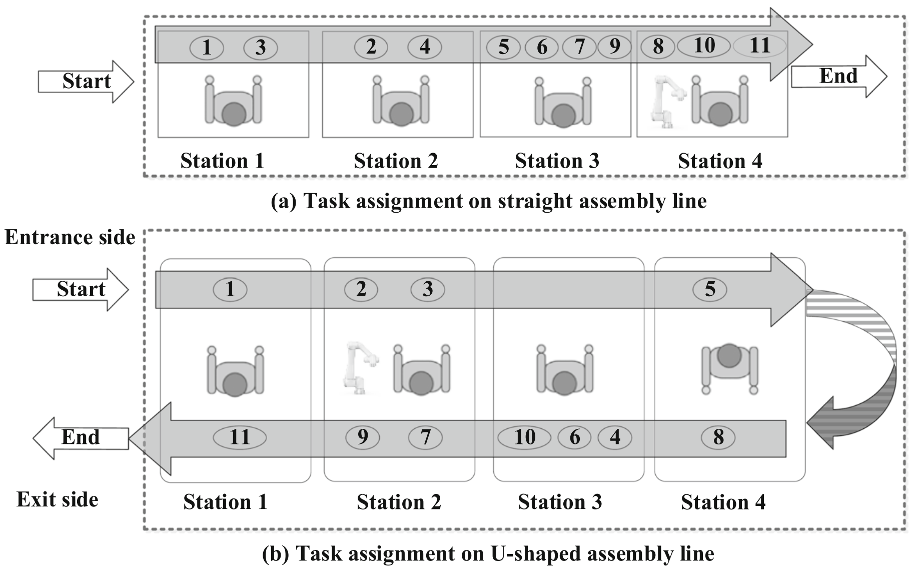

Fig. 5 Task assignment worker and robot allocation on straight assembly line and U-shaped assembly line

reduced to 8 when the budget increase to 70. Comparing the cycle times on the straight assembly line and U-shaped line, it is clear that for some situations U-shaped assembly line obtains the smaller cycle time. For instance, when the budget is 20.0, straight assembly line obtains a cycle time of 11 whereas U-shaped assembly line obtains a cycle time of 10. Figure 5 depicts the detailed task assignment worker and robot allocation on straight assembly line (Fig. 5a) and U-shaped assembly line (Fig. 5b) within the budget of 20.0. As it can be seen, the workers and cobot on the straight line operate the tasks from the first station to the last station in sequence. The U-shaped assembly line, on the contrary, is divided into entrance side and exit side. The process alternative on station 1 first operates task 1 on the entrance side and later operates task 11 on the exit side, the process alternative on station 2 first operates task 2 and task 3 on the entrance side and later operates task 7 and task 9 on the exit side, the process alternative on station 3 operates the task 4, task 6 and task 10 on the exit side in sequence, and the process alternative on station 4 first operates task 5 on the entrance side and later operates task 8 on the exist side. One clear advantage of the U-shaped assembly line is that it has lager flexibility and it might have the smaller cycle time than the straight assembly line.

6 Computational study

This section first presents the experimental design in Sect. 6.1, including the tested instances and running environments. Later, Sect. 6.2 evaluates the formulated model, where the formulated model is tested on the small-size instances. At last, Sect. 6.3 evaluates the developed algorithms and 12 other implemented algorithms statistically.

6.1 Experimental design

This study utilizes the instances consisting of 93 instances. These instances have 22 precedence diagrams, and each precedence diagram corresponds to several station numbers. The smallest-size instance has only 7 tasks and the largest-size instance has 297 tasks. For each instance, there are four types of cobots available and there are nine possible process alternatives for each task. For the operation times of tasks by process alternatives, the operation time by worker is set as the original operation times in the literature, the operation time of robots are set to be larger than that by worker, and the operation times by the collaboration are set to be smaller than that by worker.

To evaluate the performance of the proposed algorithms, they are compared with 12 algorithms. They include the late acceptance hill-climbing algorithm (LAHC)(Li et al. 2019b), simulated annealing algorithm (SA) (Li et al. 2019b), genetic algorithm (GA) (Li et al. 2019b), discrete particle swarm optimization algorithm (DPSO) (Li et al. 2019b), original cuckoo search algorithm (OCS), discrete cuckoo search algorithm (DCS)(Li et al. 2018a), original artificial bee colony algorithm(OABC), two improved artificial bee colony algorithms(ABC1 and ABC2) (Li et al. 2019b), original bee algorithm (OBA), improved bee algorithm (Çil et al. 2020) and original migrating bird optimization algorithm (OMBO).

All the implemented algorithms utilize the same encoding and decoding in Sect. 4.1 and neighbor structures presented in Sect. 4.4. And they terminate when the computation time reaches $nt \cdot nt \cdot \tau$ milliseconds (ms). And in this study the value of τ is set to 10, 20, 30, 40, 50 and 60, respectively, to observe the algorithms' performance under different running times. To have enough data for statistical

Table 4 Results by the formulated models

Instance	*ns*	LB-All	Model 1			Model 2			Model 3			Algorithms		
			UB	LB	Time	UB	LB	Time	UB	LB	Time	IABC	IMBO	Time
P7	2	12	12	12	0.12	12	12	0.08	12	12	0.08	12	12	2.94
P7	3	9	9	9	0.11	9	9	0.20	9	9	0.11	9	9	2.94
P7	4	7	7	7	0.11	7	7	0.09	7	7	0.12	7	7	2.94
P8	3	21	21	21	0.27	21	21	0.14	21	21	0.25	21	21	3.84
P8	4	17	17	17	0.47	17	17	0.31	17	17	0.45	17	17	3.84
P8	5	16	16	16	0.84	16	16	0.44	16	16	0.45	16	16	3.84
P9	3	11	11	11	0.48	11	11	0.28	11	11	0.30	11	11	4.86
P9	4	9	9	9	0.37	9	9	0.58	9	9	0.22	9	9	4.86
P9	5	8	8	8	0.90	8	8	2.20	8	8	1.50	8	8	4.86
P9	6	7	7	7	1.48	7	7	2.61	7	7	1.54	7	7	4.86
P11	3	14	14	14	0.16	14	14	0.41	14	14	0.20	14	14	7.26
P11	4	10	10	10	0.17	10	10	0.39	10	10	0.36	10	10	7.26
P11	5	9	9	9	0.25	9	9	1.34	9	9	1.31	9	9	7.26
P11	6	8	8	8	2.90	8	8	1.65	8	8	2.39	8	8	7.26
P11	7	7	7	7	0.51	7	7	0.94	7	7	0.94	7	7	7.26
P21	3	31	31	31	1.42	31	31	1.61	31	31	1.05	31	31	26.46
P21	4	24	24	24	3.32	24	24	2.57	24	24	2.50	24	24	26.46
P21	5	19	19	19	8.24	19	19	3.99	19	19	4.62	19	19	26.46
P21	6	16	16	16	7.89	16	16	7.00	16	16	4.96	16	16	26.46
P21	7	14	14	14	4.77	14	14	8.92	14	14	16.99	14	14	26.46
P21	8	13	13	13	13.40	13	13	578.70	13	13	42.04	13	13	26.46
P25	3	36	36	36	1.06	36	36	1.26	36	36	0.98	36	36	37.50
P25	4	28	28	28	20.07	28	28	5.96	28	28	8.42	28	28	37.50
P25	5	23	23	23	49.87	23	23	15.72	23	23	193.05	23	23	37.50
P25	6	19	19	19	376.79	19	19	77.70	19	19	26.86	19	19	37.50
P25	7	17	17	17	958.72	17	17	148.14	17	17	135.42	17	17	37.50
P25	8	15	15	15	174.31	15	15	204.91	15	15	387.55	15	15	37.50
P28	3	281	281	281	1.26	281	281	5.16	281	281	4.74	281	281	47.04
P28	4	221	221	221	3.81	221	221	6.88	221	221	1.56	221	221	47.04
P28	5	182	182	182	6.43	182	182	38.63	182	182	10.20	182	182	47.04
P28	6	154	154	154	15.77	154	154	12.12	154	154	16.02	154	154	47.04
P28	7	133	133	133	26.80	133	133	21.67	133	133	25.04	133	134	47.04
P28	8	117	117	117	182.18	117	117	145.08	117	117	335.35	118	118	47.04
P29	8	37	38	37	3600.00	37	37	898.00	38	37	3600.02	37	37	50.46
P29	10	31	31	31	1421.83	31	31	2057.22	31	31	1039.96	31	31	50.46
P29	12	26	26	26	1291.23	28	23	3600.02	27	24	3600.07	26	26	50.46
P29	14	23	23	23	2999.15	24	23	3600.03	24	23	3600.03	24	24	50.46
P30	7	42	42	42	42.43	42	42	235.75	42	42	177.75	42	42	54.00
P30	9	34	34	34	259.91	34	34	708.63	34	34	190.73	34	34	54.00
P30	11	27	29	26	3600.02	28	26	3600.02	29	27	3600.02	28	28	54.00
P30	13	24	24	24	375.52	24	24	776.57	24	24	335.68	24	24	54.00
P32	9	1435	1493	1337	3600.07	1486	1435	3600.02	1514	1311	3600.02	1466	1466	61.44
P32	10	1356	1400	1326	3600.02	1402	1356	3385.21	1400	1326	3600.02	1390	1390	61.44
P32	11	1193	1257	1193	3600.03	1219	1193	3600.02	1248	1193	3600.02	1210	1212	61.44
P32	12	1117	1186	1012	3600.02	1164	1117	3600.05	1182	943	3600.11	1150	1150	61.44
P35	6	72	72	72	2044.70	72	72	436.96	72	72	278.55	72	72	73.50
P35	9	50	51	50	3600.02	51	50	3600.02	51	50	3600.02	51	51	73.50

Table 4 (continued)

Instance	*ns*	LB-All	Model 1			Model 2			Model 3			Algorithms		
			UB	LB	Time	UB	LB	Time	UB	LB	Time	IABC	IMBO	Time
P35	12	40	40	40	59.47	40	40	182.11	40	40	175.64	40	40	73.50
P35	15	40	40	40	16.75	40	40	93.77	40	40	5.19	40	40	73.50
P45	4	119	119	119	54.44	119	119	62.52	119	119	26.24	120	119	121.50
P45	6	83	83	83	228.09	83	83	314.96	83	83	309.68	83	83	121.50
P45	8	63	63	63	2066.05	64	57	3600.02	63	63	2030.97	63	63	121.50
P45	10	55	55	55	12.64	55	55	16.75	55	55	7.99	55	55	121.50
P53	4	3150	3150	3150	402.56	3150	3150	1042.35	3150	3150	94.60	3150	3150	168.54
P53	6	2115	2116	2105	3600.02	2115	2115	3095.83	2115	2115	2021.67	2119	2119	168.54
P53	8	1775	1775	1775	253.34	1775	1775	54.02	1775	1775	30.97	1775	1775	168.54
P53	10	1556	1556	1556	564.14	1556	1556	1893.05	1556	1556	341.56	1556	1556	168.54
P58	6	228	229	228	3600.03	229	228	3600.02	229	228	3600.07	229	230	201.84
P58	12	105	129	105	3600.05	126	97	3600.03	127	98	3600.07	123	123	201.84
P58	18	75	88	75	3600.03	89	72	3600.10	90	70	3600.07	85	85	201.84
P58	24	46	68	46	3600.08	73	45	3600.10	78	45	3600.10	65	65	201.84
P70	8	406	409	371	3600.02	408	373	3600.08	409	406	3600.03	406	406	294.00
P70	13	206	269	206	3600.05	267	189	3600.07	265	175	3600.07	255	256	294.00
P70	18	154	194	154	3600.07	197	143	3600.10	218	143	3600.10	191	191	294.00
P70	23	110	164	110	3600.08	162	108	3600.11	160	108	3600.13	152	153	294.00

Table 5 Summarized results by models

Tested Models	#OPT	RPD-Avg	Time-Avg
Model 1	49	3.47	1100.89
Model 2	48	3.63	1140.80
Model 3	48	4.11	1069.16
IABC	48	2.76	74.51
IMBO	48	2.79	74.51

Table 6 Model statistics when solving the illustrated example in Sect. 5

Model statistics	Model 1	Model 2	Model 3
Blocks of equations	8	8	7
Single equations	66	66	53
Blocks of variables	5	5	5
Single variables	428	813	813
Nonzero elements	3,331	4,656	4,656
Discrete variables	427	812	812

analysis, all the algorithms solve the tested instances in ten repetitions and the achieved cycle times are recorded. After completing all the experiments, this study utilizes the relative percentage deviation (RPD) to transfer the achieved results with $\mathrm{RPD} = 100 \cdot (\mathrm{CT}_{\mathrm{some}} - \mathrm{CT}_{\mathrm{Best}})/\mathrm{CT}_{\mathrm{Best}}$, where CT_{some} is the cycle time by one algorithm in one run and CT_{Best} is the best cycle by all algorithms in ten runs. Here, the smaller value of RPD indicates the better performance.

The mathematical model is solved utilizing the CPLEX solver on the platform of the General Algebraic Modeling System 23.0. As the model cannot solve the large-size instance in acceptable time, only the instances with less than or equal to 70 tasks are solved by the model. And the model terminates once the optimal solution is obtained and the optimality of the achieved solution is proved or the computational time reaches 3600 s (s). The implemented algorithms are programmed with the C + + programming language of the Microsoft Visual Studio 2015. And the algorithms terminate when the given termination criterion of an elapsed computation time is satisfied. The real experiments are conducted on a set of virtual computers of a tower type of server. Each virtual computer has one virtual processor and 2 GB RAM memory and the tower type of server is equipped with two Intel Xeon E5-2680 v2 processors and 64 GB RAM memory.

6.2 Evaluating the model

This section evaluates the model by solving the small-size instances with budget of 20.0 in Table 4. In this table, LB-All in the third column is the maximum value of the lower bound or the optimal solution by the three models. UB, LB

Table 7 Average RPD values by implemented algorithms when $\tau = 40, 60$

Instance	Num	LAHC	SA	GA	DPSO	OCS	DCS	OABC	ABC1	ABC2	OBA	IBA	OMBO	IABC	IMBO
τ = 40															
P7	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P8	3	1.30	0.00	0.00	0.00	1.04	0.00	0.00	0.00	0.00	0.42	0.00	0.00	0.00	0.00
P9	4	2.27	0.00	1.14	0.00	0.45	0.45	1.14	1.82	0.68	0.68	2.27	1.14	2.05	0.23
P11	5	1.80	0.00	0.40	0.00	0.40	0.00	0.00	0.20	0.00	0.80	1.40	0.00	0.00	0.00
P21	6	1.83	0.61	1.56	0.00	2.28	0.31	0.10	1.16	0.90	2.33	1.13	0.49	0.09	0.00
P25	6	1.70	0.98	1.56	0.18	2.07	0.69	0.44	1.18	1.09	1.48	1.57	0.58	0.56	0.09
P28	6	1.31	0.52	0.92	0.77	1.37	0.44	0.65	0.62	0.52	1.34	1.04	0.47	0.29	0.40
P29	4	2.06	2.12	3.31	2.44	4.84	2.19	2.62	2.19	2.36	5.35	2.28	2.52	1.33	1.09
P30	4	0.80	0.43	2.30	1.50	3.65	0.10	1.63	0.62	0.34	3.68	0.76	1.42	0.00	0.00
P32	4	1.39	1.21	1.76	1.07	2.07	0.69	1.21	1.03	0.82	2.08	1.08	0.93	0.27	0.32
P35	4	0.56	0.10	0.65	0.17	1.09	0.08	0.07	0.29	0.20	5.73	0.21	0.08	0.00	0.00
P45	4	0.78	0.48	1.09	0.87	1.32	0.62	0.79	0.80	0.62	0.98	1.08	0.91	0.42	0.36
P53	4	0.80	0.35	0.39	0.04	0.54	0.31	0.06	0.37	0.29	1.74	0.92	0.30	0.28	0.04
P58	4	0.90	0.88	2.51	2.47	3.48	1.10	2.77	1.10	1.00	3.33	1.02	1.72	0.65	1.38
P70	4	0.74	0.77	1.83	1.53	3.13	0.87	1.80	0.94	0.86	2.50	0.77	1.40	0.44	0.58
P75	4	0.30	0.24	1.47	1.31	1.89	0.28	1.59	0.30	0.43	1.84	0.20	0.83	0.00	0.43
P83	4	0.64	0.52	1.74	1.15	2.25	0.75	1.53	0.45	0.70	1.92	0.75	0.59	0.17	0.34
P89	4	0.00	0.00	1.52	0.75	2.23	0.00	1.25	0.00	0.00	1.88	0.00	0.00	0.00	0.00
P94	4	0.66	0.37	2.13	1.84	2.81	0.87	1.87	0.79	0.81	2.24	0.82	0.99	0.59	0.91
P111	4	0.55	0.53	2.11	1.66	3.51	0.76	2.12	0.72	0.58	2.74	0.41	1.70	0.56	0.52
P148	4	0.73	0.71	1.21	0.66	1.42	0.51	0.86	0.64	0.50	0.77	0.62	0.20	0.23	0.31
P297	4	0.23	0.22	1.77	1.87	2.22	0.49	1.95	0.40	0.41	1.86	0.24	1.05	0.29	0.68
Avg		1.03	0.52	1.44	0.89	2.01	0.53	1.08	0.74	0.62	2.08	0.89	0.78	**0.37**	**0.34**
τ = 60															
P7	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P8	3	1.30	0.00	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.21	0.00	0.00	0.00	0.00
P9	4	2.27	0.00	1.14	0.00	0.23	0.00	0.68	1.36	0.23	0.68	2.27	1.14	2.05	0.23
P11	5	1.80	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.00	0.80	1.40	0.00	0.00	0.00
P21	6	1.83	0.61	1.35	0.00	1.79	0.31	0.00	1.16	0.71	2.33	1.13	0.49	0.09	0.00
P25	6	1.70	0.98	1.51	0.00	1.79	0.60	0.18	1.03	1.09	1.38	1.57	0.49	0.32	0.00
P28	6	1.26	0.51	0.87	0.74	1.25	0.44	0.55	0.54	0.47	1.29	1.03	0.45	0.19	0.37
P29	4	1.87	2.02	3.31	2.44	4.76	2.19	2.44	2.19	2.30	5.35	2.28	2.52	1.02	1.09
P30	4	0.80	0.36	2.24	1.04	3.65	0.10	1.44	0.44	0.34	3.68	0.57	1.42	0.00	0.00
P32	4	1.39	1.18	1.57	1.03	2.03	0.59	1.16	0.92	0.77	2.08	1.03	0.90	0.24	0.23
P35	4	0.56	0.10	0.60	0.10	1.06	0.08	0.03	0.29	0.20	5.73	0.21	0.08	0.00	0.00
P45	4	0.78	0.48	1.09	0.83	1.28	0.62	0.79	0.80	0.62	0.98	1.08	0.91	0.42	0.31
P53	4	0.80	0.35	0.38	0.03	0.47	0.30	0.06	0.32	0.29	1.72	0.92	0.30	0.28	0.00
P58	4	0.85	0.84	2.40	2.25	3.33	1.02	2.72	1.10	0.92	3.33	0.94	1.68	0.59	1.30
P70	4	0.61	0.68	1.75	1.45	2.94	0.76	1.70	0.89	0.77	2.50	0.71	1.40	0.37	0.55
P75	4	0.28	0.24	1.47	1.21	1.79	0.25	1.49	0.30	0.38	1.84	0.20	0.73	0.00	0.42
P83	4	0.64	0.51	1.65	1.00	2.20	0.66	1.45	0.44	0.66	1.92	0.74	0.55	0.14	0.33
P89	4	0.00	0.00	1.42	0.38	2.13	0.00	1.13	0.00	0.00	1.88	0.00	0.00	0.00	0.00
P94	4	0.60	0.30	2.02	1.78	2.70	0.79	1.78	0.69	0.77	2.24	0.74	0.94	0.55	0.89
P111	4	0.45	0.43	1.99	1.56	3.26	0.64	2.05	0.61	0.52	2.74	0.36	1.70	0.46	0.43
P148	4	0.73	0.70	1.14	0.61	1.34	0.48	0.79	0.63	0.50	0.76	0.61	0.20	0.23	0.25
P297	4	0.18	0.15	1.73	1.77	2.14	0.42	1.88	0.33	0.37	1.83	0.20	1.02	0.23	0.66
Avg		1.00	0.49	1.35	0.80	1.87	0.47	0.97	0.66	0.56	2.06	0.87	0.76	**0.32**	**0.31**

Table 8 Overall average RPD values under different computation times

Algorithm	τ = 10	τ = 20	τ = 30	τ = 40	τ = 50	τ = 60
LAHC	1.130	1.078	1.045	1.028	1.009	0.998
SA	0.606	0.548	0.533	0.520	0.506	0.494
GA	1.780	1.589	1.489	1.440	1.392	1.354
DPSO	1.269	1.069	0.964	0.893	0.846	0.799
OCS	2.477	2.216	2.111	2.012	1.918	1.872
DCS	0.778	0.648	0.565	0.527	0.497	0.471
OABC	1.480	1.248	1.128	1.077	1.008	0.975
ABC1	1.022	0.873	0.797	0.738	0.712	0.664
ABC2	0.857	0.713	0.648	0.618	0.586	0.561
OBA	2.156	2.129	2.085	2.079	2.071	2.060
IBA	1.001	0.959	0.914	0.895	0.880	0.870
OMBO	0.953	0.826	0.795	0.778	0.768	0.758
IABC	0.567	0.438	0.399	0.373	0.331	0.322
IMBO	0.580	0.432	0.363	0.340	0.319	0.311

and Time are the upper bound of the cycle time, the lower bound of the cycle time and the consumed computation time, respectively. The last three columns present the best results by IABC, the results by IMBO in ten repetitions and the utilized computation time ($nt \cdot nt \cdot 60$ ms). Here, one model is capable of solving one instance optimally within the given computation time when UB is equal to the LB. As you can see, the models can solve the small-size instances optimally with less than or equal to 28 tasks, whereas they might not obtain the optimal solution for the remained instances. The proposed algorithms are also achieving the optimal solutions for the small-size instances with less than or equal to 28 tasks, and they might achieve clear superiority over the mathematical models when solving the large-size instances. For instances, IABC and IMBO outperform Model 1, Model 2 and Model 3 when solving the P70 with 13, 18 and 23 stations.

To have a better observation of the models' performance, Table 5 presents the numbers the instances which are solved optimally (#OPT), the average value of the RPD values (RPD-Avg), the average computation time of all the instances (Time-Avg). As the lower bounds can be achieved by the models, here RPD is calculated with $\mathrm{RPD} = 100 \cdot (\mathrm{CT}_{\mathrm{some}} - \mathrm{CT}_{\mathrm{LB}})/\mathrm{CT}_{\mathrm{LB}}$, where CT_{some} is the cycle time by one method and CT_{LB} is the maximum value of the lower bounds by three models. As you can see, among the three models, Model 1 is the best performer in terms of the #OPT and RPD-Avg; Model 3 is the best performer in terms of the Time-Avg. For the comparison between the models and algorithms, it is clear that IABC and IMBO obtains the smaller values of RPD-Avg within the short computation time, demonstrating that the algorithms outperform the MILP models when solving the large-size instances in term of the solution quality.

To further analyze the reasons leading to the different performances of the models, Table 6 provides the model statistics when solving the illustrated example in Sect. 5 as an example. As you can see, Model 1 has the smallest number of single variables, the smallest number of nonzero elements and the smallest number of discrete variables, and hence Model 1 achieves the best performance in terms of the #OPT and RPD-Avg. Model 3 has the smallest number of blocks of equations and the smallest number of single equations, and hence Model 3 achieves the best performance in terms of the Time-Avg.

6.3 Evaluating the implemented algorithms

As the parameter values have great impact on the algorithms' performance, the parameters of each algorithm are calibrated before running the final experiments utilizing the

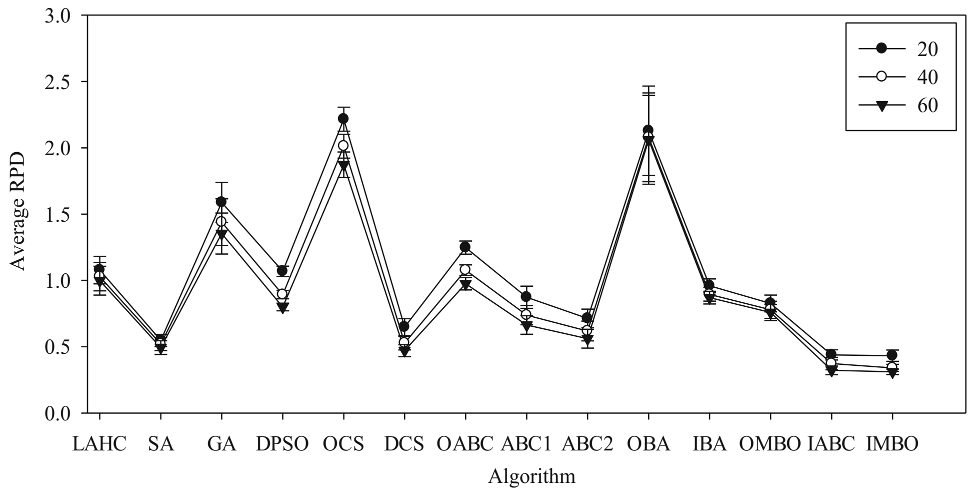

Fig. 6 Means plot and 95% Tukey HSD confidence intervals for the interactions between algorithms and termination criteria

Design of Experiments approach and multi-factor analysis of variance (ANOVA) technique. Firstly, two or three values are selected for each parameter with other parameter fixed. Secondly, all the combinations of the parameter values solve a set of 10 instances with different sizes for 10 times with an elapsed computation time of $nt \cdot nt \cdot 60$ *ms*. Finally, ANOVA test is conducted to select the best parameter values where the parameters are selected as controlled factors and the average RPD of all the instances in one run is selected as response variable. This parameter calibration method has been widely in literature (see (Li et al. 2019b), (Çil et al. 2020) and many others) and hence the detailed information are not provided here for space reasons. After parameter calibration, the selected parameter values of the IABC are as follows: population size is 5, parameter *limit* is 500, and parameter r is 0.1. And in the modified scout phase, this algorithm obtains 10 neighbor solutions of one abandoned solution by conducting the neighbor operators for two times. The selected parameter values of the IMBO are as follows: population size is 5, parameter k is 11, parameter x is 5, parameter m is 20, parameter T_{init} is 0.5, parameter α is 0.95 and parameter rt is 500.

After conducting all the experiments, the achieved cycle times are transferred into RPD values. As 10 runs and six computation times are conducted, there are a number of $93 \cdot 10 \cdot 6$ data to evaluate one algorithm statistically. Table 7 provides the average RPD by implemented algorithms when τ is equal to 40 and 60. In this table, Instance donates the size of the instances, where the number is the number of tasks. Num. presents the number of cases with different station numbers. The detailed results of the cycle times in each run are not exhibited due to space limit, and they are available upon request. As you can see, under $\tau = 40$, IMBO is the best performer with the overall average RPD of 0.34, IABC is the second-best performer with the overall average RPD of 0.37, and SA is the third-best performer with the overall average RPD of 0.52. When $\tau = 60$, IMBO is also the best performer, IABC is the second-best performer and SA is the third-best performer again.

To have a direct observation of the algorithms' performance under different computation times, Table 8 provides the overall average RPD values under τ = 10, 20, 30, 40, 50 and 60, respectively. It is observed that all the algorithms produce improvements when the computation time increases. Meanwhile, the proposed IMBO and IABC are the two best performers under all the termination criteria, demonstrating the superiority of the IMBO and IABC over the remained algorithms.

This study also conducts the statistical analysis to make sure that the observed differences are statistically significant. The ANOVA test is again utilized to analyze the results here, where algorithms and termination criteria are the controlled factors and the average RPD in one run is the response variable. However, the initial ANOVA test shows that normality of the residuals is not satisfied. In fact, this situation lies behind that algorithms have quite different performances, and the normality of the residuals can be satisfied if we only consider the best three algorithms. Meanwhile, the nonparametric Friedman test is also applied to ascertain the findings by the ANOVA test. As the Friedman test can only consider one controlled factor, the Friedman test is conducted for six times utilizing the different results under six termination criteria. The ANOVA test shows that there is statistically significant difference between the algorithms, the termination criteria and the interactions of the two controlled factors. Friedman test shows that there is statistically significant difference between the algorithms under all the computation times. Figure 6 illustrates the Means plot and 95% Tukey HSD confidence intervals for the interactions. Clearly, IABC and IMBO are the two best performers under all the computation time and they outperform the remained algorithms statistically. All in all, the comparative study and the statistical analysis verify the superiority of the IMBO and IABC over the compared algorithms, and, as a consequence, they could be regarded as the effective and powerful methods in solving the considered problem.

7 Conclusions and future research

Usage of collaborative robot (cobot) gains more and more applications in recent days to assist the human workers in the assembly line. This research studies the U-shaped assembly line balancing problem with cobots to minimize the cycle time, where several cobots of different types are selected under the budget constraint. Three mixed-integer linear programming (MILP) models are developed to formulate this new problem, and these models are capable of solving the small-size instance optimally. In addition, this study develops two algorithms, improved artificial bee colony algorithm and improved migrating bird optimization algorithm to tackle the large-size instance effectively. The two algorithms utilize the new encoding scheme and decoding procedure to obtain feasible solution and propose several improvements to enhance the exploitation and exploration capacity.

A set of experiments are conducted to evaluate the formulated models and developed algorithms. The experimental results demonstrate that the utilization of collaborative robots helps reduce the cycle time. And the comparative study between the models shows that the models can solve the small-size instances optimally, whereas they cannot obtain satisfying solutions for the large-size instances within the acceptable computation time. The statistical analysis between the algorithms demonstrates that proposed algorithms are the two best performers and they outperform the other 12 implemented methodologies.

The findings of this study can be utilized in the U-shaped assembly line to obtain the smaller cycle time and the developed algorithm might be integrated into the decision support system to achieve high-quality solution to assist the line manager to design the U-shaped assembly lines. Future research stems from developing the branch, bound and remember algorithm to solve the large-size instance optimally to extending the considered problems. This considered problem might be extended by considering the cobot breakdown, the mixed-model U-shaped assembly line, and the multiple objectives utilizing the multi-objective algorithms.

Acknowledgements This project is partially supported by National Natural Science Foundation of China under grants 62173260 and 61803287 and China Postdoctoral Science Foundation under grant 2018M642928. The authors are grateful for the valuable comments by the anonymous referees to improve this paper.

Funding The authors have not disclosed any funding. National Natural Science Foundation of China, 61803287, 51875421, Postdoctoral Research Foundation of China, 2018M642928, Zixiang Li.

Data availability The data may be available from the authors upon request.

Declarations

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

Human and animal rights The authors did not use any humans and animals in this research work.

References

Avikal S, Jain R, Mishra PK, Yadav HC (2013) A heuristic approach for U-shaped assembly line balancing to improve labor productivity. Comput Ind Eng 64(4):895–901. https://doi.org/10.1016/j.cie.2013.01.001

Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solution approaches. Int J Prod Econ 142(2):259–277

Baykasoglu A, Dereli T (2009) Simple and U-type assembly line balancing by using an ant colony based algorithm. Math Comput Appl 14(1):1–12

Çil ZA, Li Z, Mete S, Özceylan E (2020) Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration. Appl Soft Comput 93:106394. https://doi.org/10.1016/j.asoc.2020.106394

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

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

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. Computers Industr Eng 139:106182. https://doi.org/10.1016/j.cie.2019.106182

Erel E, Sabuncuoglu I, Aksu BA (2001) Balancing of U-type assembly systems using simulated annealing. Int J Prod Res 39(13):3003–3015. https://doi.org/10.1080/00207540110051905

Fattahi A, Turkay M (2015) On the MILP model for the U-shaped assembly line balancing problems. Eur J Oper Res 242(1):343–346. https://doi.org/10.1016/j.ejor.2014.10.036

Hashemi-Petroodi SE, Thevenin S, Kovalev S, Dolgui A (2020) Operations management issues in design and control of hybrid human-robot collaborative manufacturing systems: a survey. Annu Rev Control 49:264–276. https://doi.org/10.1016/j.arcontrol.2020.04.009

Hwang RK, Katayama H, Gen M (2008) U-shaped assembly line balancing problem with genetic algorithm. Int J Prod Res 46(16):4637–4649. https://doi.org/10.1080/00207540701247906

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

Koltai T, Dimény I, Gallina V, Gaal A, Sepe C (2021) An analysis of task assignment and cycle times when robots are added to human-operated assembly lines, using mathematical programming models. Int J Prod Econ 242:108292. https://doi.org/10.1016/j.ijpe.2021.108292

Krüger J, Lien TK, Verl A (2009) Cooperation of human and machines in assembly lines. CIRP Ann 58(2):628–646. https://doi.org/10.1016/j.cirp.2009.09.009

Li Z, Dey N, Ashour AS, Tang Q (2018a) 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

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

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

Li Z, Janardhanan MN, Tang Q (2021) Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots. Neural Comput Appl 33(14):8575–8596. https://doi.org/10.1007/s00521-020-05610-2

Li Z, Janardhanan MN, Tang Q, Ponnambalam SG (2019) Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. Swarm Evolutionary Comput 50:100567. https://doi.org/10.1016/j.swevo.2019.100567

Li Z, Kucukkoc I, Tang Q (2017) New MILP model and station-oriented ant colony optimization algorithm for balancing U-type assembly lines. Comput Ind Eng 112:107–121. https://doi.org/10.1016/j.cie.2017.07.005

Li Z, Kucukkoc I, Zhang Z (2018b) Branch, bound and remember algorithm for U-shaped assembly line balancing problem. Comput Ind Eng 124:24–35. https://doi.org/10.1016/j.cie.2018.06.037

Miltenburg GJ, Wijngaard J (1994) The U-line line balancing problem. Manage Sci 40(10):1378–1388

Nilakantan JM, Ponnambalam S (2016) Robotic U-shaped assembly line balancing using particle swarm optimization. Eng Optim 48(2):231–252

Nourmohammadi A, Fathi M, Ng AHC (2022) Balancing and scheduling assembly lines with human-robot collaboration tasks. Computers Op Res 140:105674. https://doi.org/10.1016/j.cor.2021.105674

Oksuz MK, Buyukozkan K, Satoglu SI (2017) U-shaped assembly line worker assignment and balancing problem: a mathematical model and two meta-heuristics. Comput Ind Eng 112:246–263. https://doi.org/10.1016/j.cie.2017.08.030

Özcan U, Toklu B (2009) A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems. J Intell Manuf 20(1):123–136. https://doi.org/10.1007/s10845-008-0108-2

Rabbani M, Behbahan SZB, Farrokhi-Asl H (2020) The collaboration of human-robot in mixed-model four-sided assembly line balancing problem. J Intell Rob Syst 100(1):71–81. https://doi.org/10.1007/s10846-020-01177-1

Sabuncuoglu I, Erel E, Alp A (2009) Ant colony optimization for the single model U-type assembly line balancing problem. Int J Prod Econ 120(2):287–300. https://doi.org/10.1016/j.ijpe.2008.11.017

Samouei P, Ashayeri J (2019) Developing optimization and 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

Scholl A, Klein R (1999) ULINO: Optimally balancing U-shaped JIT assembly lines. Int J Prod Res 37(4):721–736. https://doi.org/10.1080/002075499191481

Urban TL, Chiang W-C (2006) An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. Eur J Op Res 168(3):771–782. https://doi.org/10.1016/j.ejor.2004.07.027

Weckenborg C, Kieckhäfer K, Müller C, Grunewald M, Spengler TS (2020) Balancing of assembly lines with collaborative robots. Bus Res 13:93–132. https://doi.org/10.1007/s40685-019-0101-y

Weckenborg C, Spengler TS (2019) Assembly line balancing with collaborative robots under consideration of ergonomics: a cost-oriented approach. IFAC-PapersOnLine 52(13):1860–1865. https://doi.org/10.1016/j.ifacol.2019.11.473

Weckenborg C, Thies C, Spengler TS (2022) Harmonizing ergonomics and economics of assembly lines using collaborative robots and exoskeletons. J Manuf Syst 62:681–702. https://doi.org/10.1016/j.jmsy.2022.02.005

Yaphiar S, Nugraha C, and Ma'ruf A (2020). Mixed Model Assembly Line Balancing for Human-Robot Shared Tasks. Paper presented at the International Manufacturing Engineering Conference and The Asia Pacific Conference on Manufacturing Systems 2019, Singapore.

Zhang Z, Tang Q, Han D, Li Z (2019) Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment. Neural Comput Appl 31(11):7501–7515. https://doi.org/10.1007/s00521-018-3596-9

Zhang Z, Tang Q, Li Z, Zhang L (2019b) 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

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. Computers Op Res 118:104905. https://doi.org/10.1016/j.cor.2020.104905

Zhang Z, Tang Q, Zhang L (2019c) 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