Computers & Industrial Engineering 150 (2020) 106862

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Multi-manned assembly line balancing with time and space constraints: A MILP model and memetic ant colony system

Zikai Zhang [a,b], Qiuhua Tang [a,b,*], Manuel Chica [c,d]

[a] Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, China
[b] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, China
[c] Andalusian Research Institute DaSCI "Data Science and Computational Intelligence", University of Granada, 18071 Granada, Spain
[d] School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW 2308, Australia

ARTICLE INFO

Keywords:
Time and space assembly line balancing
Multi-manned stations
MILP model
Ant colony optimization
Memetic algorithms

ABSTRACT

In the automotive and electronics industries, more than one operator work in the same workstation to assemble a high volume of products. When assigning the tasks of these products to workstations, we should fulfill the cycle time and precedence relationships. Limited research has investigated space restrictions to store tools or components (i.e., time and space assembly line balancing problem) but without multi-manned workstations. Therefore, this paper addresses the time and space assembly line balancing problem with multi-manned workstations. Our model includes five kinds of constraints by considering task assignment, precedence, cycle time, sequencing and space constraints. Our aim is to minimize the total number of workstations and operators via a new MILP model and memetic ant colony system. The memetic ant algorithm uses a new solution generation method which integrates 16 heuristic rules to help each ant of the algorithm to effectively build a feasible solution. New pheromone release strategies, including deposition and evaporation, are employed to update the global pheromone quantity. Additionally, a new best solution update method does not retain the solution with minimum objective function but balances the workload of each operator. Our experiments show the effectiveness of solving the MILP model by exact methods in small-scaled instances and the superiority of the memetic ant colony optimization algorithm in all the instances.

1. Introduction

Assembly lines are widely used as the primary production model in manufacturing industries, such as automobiles, food, toys, and furniture. An assembly line contains a sequence of workstations to perform a set of tasks. The assembly line balancing problem (ALBP) (Scholl & Becker, 2006) models the optimization of an assembly line by assigning these tasks into the stations and by fulfilling some constraints such as task assignment constraint, precedence relation constraint, and cycle time constraint to optimize one or more objective functions. The task assignment constraint requires that every task must be assigned to exactly one station. The precedence relation constraint states that a task is assigned to a station after all its predecessors complete the assignment. The cycle time constraint implies that the processing time of each station does not exceed the cycle time. ALBP can be divided into four categories according to the objective function (Boysenaab, 2007): ALBP-1 minimizes the number of stations for a fixed cycle time; ALBP-2 minimizes the cycle time for a given number of stations; ALBP-E maximizes the line efficiency without knowing the cycle time and number of stations; and ALBP-F finds a feasible solution when the both are known.

The traditional ALBP simplifies the real-world industrial problem. It assumes that an assembly line mass produces one homogeneous product and is a one-sided serial line layout within single man stations, etc. (Roshani, Roshani, Roshani, Salehi, & Esfandyari, 2013). These features may be unrealistic in real industries. Hence, in recent years, researchers paid more attention to more realistic problems by trying to consider additional characteristics into ALBP. Examples are mixed-model (Mosadegh, Ghomi, & Süer, 2019; Samouei & Ashayeri, 2019), multi-manned (Lopes, Pastre, Michels, & Magatão, 2019; Michels, Lopes, Sikora, & Magatão, 2019), space constraint (Chica, Bautista, Cordón, & Damas, 2016) and stochastic processing time (Mosadegh et al., 2019). All the latter problem formulations are referred as general assembly line balancing problems (GALBPs).

The multi-manned assembly line balancing problem (MALBP)

* Corresponding author.
E-mail addresses: zhangzikai0703@gmail.com (Z. Zhang), tangqiuhua@wust.edu.cn (Q. Tang), manuelchica@ugr.es (M. Chica).

https://doi.org/10.1016/j.cie.2020.106862
Received 12 February 2020; Received in revised form 28 July 2020; Accepted 15 September 2020
Available online 10 October 2020

belongs to GALBP. It allows more than one operator to process multiple tasks in the same stations. The admitted maximum number of operators in each multi-manned workstation is predefined by the system designer according to the real production. The MALBP usually models those industries with large size and high volume of products such as automotive industry (Dimitriadis, 2006). MALBP is more preferable and realistic than the traditional ALBP because it reduces the length of assembly line, the amount of throughput time, the cost of tools and fixtures, the material handling, worker movement, and setup time (Fattahi, Roshani, & Roshani, 2010). Hence, this paper aims to study the MALBP with the minimization of the total workstations and operators.

However, existing research assumed that each task just has one temporal attribute and the cycle time constraint is related to the temporal attribute. Bautista and Pereira (Bautista & Pereira, 2007) investigated the specific characteristics of the Nissan automotive plant and found that different tasks require different tools and components during assembling; and each station in the assembly line should have enough space to store the tools and components. Hence, each task has an additional spatial attribute. The consideration of space constraint in assembly line (the time and space assembly line balancing problem, TSALBP) achieves a better modelling of real-world assembly scenarios and has become an important topic in assembly line balancing (Chica, Cordón, Damas, & Bautista, 2012).

This paper mainly makes two contributions to existing literature. Firstly, we propose a MILP model to formulate the multi-manned assembly line balancing problems with both time and space constraints. This MILP model involves one objective with the minimization of the number of total workstations and operators and five kinds of constraints including task assignment, precedence, cycle time, sequencing and space constraints. With respect to the existing models in the literature, this MILP model first considers the multi-manned and space attributes simultaneously. Secondly, even simple versions of TSALBP are known as a NP-hard problem (Karp, 2010) and therefore, it is time-consuming to obtain the optimal solutions when increasing the size of the problem. In this work, an ant colony system (ACS) (Dorigo, Maniezzo, & Colorni, 1996) is extended here to tackle the middle- and large-scaled problem. The use of constructive procedures such as ACS is more appropriate than non-constructive metaheuristics to solve problems with many constraints such as the TSALBP (Chica, Cordón, Damas, & Bautista, 2010). We propose a memetic ACS algorithm, called ACS_LS, with a new solution generation designed to improve the efficiency and reduce the idle time. 16 heuristic rules are embedded to determine the best constructive decision and pheromone values and further enhance its performance. The included *ad-hoc* local search is integrated to enhance the exploitation phase of the metaheuristic.

In order to validate our study, 154 benchmark instances drive from 10 calibration instances and 144 comparison instances are employed respectively to determine the best parameter combination of ACS_LS and test the performance of MILP model and algorithm. Specifically, we first compare the proposed ACS_LS with the results obtained by the GAMS/CPLEX to illustrate its effectiveness in small-scale instances. Then, 16 heuristic rules embedded in the memetic ACS algorithm are compared with each other to find the best one. Finally, we analyze the effectiveness of the local search in ACS.

The remainder of this paper is structured as follows. Section 2 reports the literature review. Section 3 defines the novel mixed-integer linear programming model. Section 4 details the proposed memetic ACS and Section 5 reports the experimentation results. Section 6 concludes with the most important insights and future work.

2. Literature review

This section summarizes the existing research related to our assembly line balancing problem. We first present a review on general assembly line balancing in Section 2.1. Later, in Section 2.2., we report the latest developments of multi-manned assembly line balancing. Finally, in Section 2.3, we focus on the time and space assembly line balancing problem by also highlighting the novelty of this research with respect to the existing literature.

2.1. The general assembly line balancing problems

In industrial manufacturing there are very different conditions occurring in assembly line balancing problems, which results in the multifaceted ALBPs. These problems are collectively known as GALBPs. They are more in line with the actual mode of production compared with the simple ALBP. The GALBPs that receive the most attention nowadays are: U-shaped assembly line balancing, mixed-model assembly line balancing, robotic assembly line balancing, two-sided assembly line balancing and multi-manned assembly line balancing, etc.

The U-shaped assembly line balancing assigns the tasks into the entrance of the workstations after the complete allocation of their predecessors or into the exit of the workstations after the complete allocation of their successors. This important characteristic makes the U-shaped lines become one of the most important components for a successful implementation of just-in-time production (Miltenburg, 2001). Recently, Oksuz, Buyukozkan, and Satoglu (2017) and Zhang, Tang, Han, and Li (2018) introduced the worker assignment into U-shaped assembly line balancing and defined the task time depending on the worker, and then respectively designed artificial bee colony, genetic algorithm and enhanced migrating birds optimization to tackle this new problem. Babazadeh and Javadian (2018) considered the U-shaped line into uncertain circumstances and proposed a modified NSGA-II to optimize two conflicting objectives: the set of constraints and the line efficiency. More works can be found in Şahin and Kellegöz (2016), Li et al. (2017), Nourmohammadi et al. (2019).

With respect to mixed-model assembly line balancing, models with different features need to be assembled in a single line and the difference task time from different models leads to the risk of exceeding the cycle time and results in utility work (Kim, Kim, & Kim, 2000). In order to address this problem, several meta-heuristic algorithms have been developed: artificial bee colony (Buyukozkan, Kucukkoc, Satoglu, & Zhang, 2016), ACO (Kucukkoc & Zhang, 2016), genetic algorithm (GA) (Zhao, Hsu, Chang, & Li, 2016) and particle swarm optimization (PSO) (Delice, Kızılkaya Aydoğan, Özcan, & İlkay, 2017), etc.

As for robotic assembly line balancing, it assigns both tasks and robots into workstations, and the task time differs from the robots. Recently, manufacturing enterprises begins to apply green manufacturing to workshop production, which leads to more focus on reducing energy consumption. Mukund Nilakantan, Huang, and Ponnambalam (2015) investigated the minimization of energy consumption in robotic assembly line balancing and proposed a particle swarm optimization to solve it. Li, Tang, and Zhang (2016) aimed at the robotic two-sided assembly line balancing and developed a restart simulated annealing algorithm (SA) to optimize the cycle time and energy consumption simultaneously. Zhang, Tang, Li, and Zhang (2018) mainly studied the U-shaped robotic assembly line balancing and designed multi-objective artificial bee colony to tackle this problem. Based on this research, Zhang, Tang, and Zhang (2019) further investigated the carbon and noise emission in U-shaped robotic assembly line balancing and proposed a novel model and meta-heuristics to minimize the carbon and noise emission and cycle time concurrently.

In above GALBPs, they just employed one operator or robot in each workstation to assemble these tasks; while with the scale of the complex products increasing, the simple line would need hundreds of workstations, which further increases the space occupancy and brings huge production costs. In this case, multi-manned assembly line is designed where operators at the same workstation can process multiple tasks simultaneously.

2.2. The multi-manned assembly line balancing problems

The MALBP was brought to the attention of the academia by Dimitriadis (2006). After that, Fattahi et al. (2010) developed a mixed integer mathematical model and an ant colony optimization algorithm to minimize the total number of workers as the first objective as well as the number of workstations as the second one. Kellegöz and Toklu (2012) proposed an efficient branch and bound algorithm (B&B) combined a branching scheme and some dominance and feasibility criteria for the MALBP. Roshani et al. (2013) designed a simulated annealing algorithm to optimize the line efficiency, line length and smoothness index of MALBP. Kellegöz and Toklu (2015) developed another new mixed integer programming model and presented a priority rule-based genetic algorithm to minimize the number of workers. Kellegöz (2016) designed a Gantt based heuristic simulated annealing algorithm to tackle type-I multi-manned assembly line balancing problem. Roshani and Giglio (2016) proposed a mixed integer mathematical formulation to minimize the cycle time and solved this problem by two methods, respectively named as ISA and DSA. Chen (2017) considered the task, operator and workstation allocations and designed a simulated annealing algorithm to minimize the number of operators and stations. Li, Wang, and Yang (2018) aimed at type-II multi-manned assembly line balancing problem and developed a simulated annealing algorithm where a new solution generation and search methods are considered to minimize the cycle time. Michels et al. (2019) presented a new mixed-integer linear model with strong symmetry break constraints and decomposed this problem into a new Benders' decomposition algorithm to achieve the minimization of the number of operators in large-scaled problems.

Apart from the original MALBP, this problem has been also extended in several aspects. For instance, Yilmaz and Yilmaz (2015) considered load-balancing constraints into the multi-manned assembly line balancing problem. Roshani and Ghazi Nezami (2017) introduced and modeled mixed-model on multi-manned assembly line balancing problem, and developed a simulated annealing algorithm to minimize the total number of operators and stations. Chen, Cheng, and Li (2018) proposed a mixed integer programming model for the MALBP with resources constraint. They assumed that production in the factory requires the use of specialized machinery and equipment, tools, and operators with specialized skills. Hence, when assigning the tasks into stations, these resources also were allotted in stations. To further achieve the optimization, a hybrid heuristic approach combined genetic algorithm was also proposed in the latter reference. Lopes et al. (2019) studied flexible multi-manned assembly line balancing problem and designed a novel model-based heuristic procedure to shorter the line length. Şahin and Kellegöz (2019) investigated the resource investment and balancing of multi-manned assembly lines and adapted a particle swarm optimization algorithm hybrid a special constructive heuristic to reduce the cost of workstations and required renewable resources.

2.3. Time and space assembly line balancing problems

Bautista and Pereira (2007) first proposed the space constraint into assembly line balancing problem by. They named this new problem as time and space assembly line balancing problem (TSALBP) and proposed a mathematical model and an ant colony optimization algorithm to solve it. In addition, they also divided this problem into eight categories depending on the variable to optimize (number of stations, the cycle time, and/or the allowed space information). Since then, Chica et al. (2010) focused on the1/3 multi-objective variant of TSALBP and designed an ant colony optimization and random greedy research to optimize the number and area of stations. Bautista and Pereira (2011) proposed an adaptation of the bounded dynamic programming to solve TSALBP-1. In this method, they designed different lower bounds to assess the quality of the obtained solutions. Later, more multi-objective meta-heuristics were developed to solve the time and space assembly line balancing problem, such as genetic algorithm (Chica, Cordón, & Damas, 2011), memetic algorithm (Chica et al., 2012), ant colony algorithm (Rada-Vilela, Chica, Cordón, & Damas, 2013) and evolutionary algorithm (EA) (Chica et al., 2016).

Note that in assembly line balancing, generally, there are three solving methods: exact, heuristic and meta-heuristic (Zhang et al., 2018). The specific constraints (such as precedence relation, multi-manned and space constraints) makes the problem strong NP-hard, and the exact methods are not usually appropriate in solving large-scale problems, and they even cannot obtain feasible solutions. The heuristics are convenient to be employed but sometimes they cannot obtain satisfactory solutions. The meta-heuristics are reported to have achieved ideal solutions despite scale limitation on site. Hence, we have selected a metaheuristic (i.e., ACO) due to the strong NP-hard nature of this new problem and the convenience of solving it through meta-heuristics. The specific search mechanism of an ACO can find a new solution guided by the local and global pheromone and in a constructive way, ideal for problems with many constraints as ours. The memetic variant of ACO and local search also ensures a convenient local and global search ability. This algorithm has shown great performance in tackling related works. For example, Fattahi et al. (2010) used ACO solve multi-manned assembly line balancing; Bautista and Pereira (2011) and Chica et al. (2010) respectively used ACO to solve single- and multi-objective time and space assembly line balancing. Hence based on the above researches, this paper uses a memetic variant of ACO where three improvements are included to tackle assembly line balancing problem with multi-manned and space constraints.

Through the above survey, we summarize all the related literature in Table 1 (multi-manned or space constraints). From the study we see there is a lack of research on the multi-manned assembly line balancing problem with time and space constraints, which is the motivation of this work. The main contribution of our work to the existing literature is the following:

Table 1
Publications about multi-manned and/or space constraints in assembly line balancing.

Reference	Attribute		Model	Lower bound	Meta-heuristic
	Multi-manned	Space			
Fattahi et al. (2010)	√		√		ACO
Kellegöz and Toklu (2012)	√			√	B&B
Roshani et al. (2013)	√				SA
Kellegöz and Toklu (2015)	√		√	√	GA
Kellegöz (2016)	√		√	√	SA
Roshani and Giglio (2016)	√		√	√	SA
Chen (2017)	√		√		SA
Li et al. (2018)	√				SA
Michels et al. (2019)	√		√		–
Yilmaz and Yilmaz (2015)	√		√		–
Roshani and Ghazi Nezami (2017)	√		√	√	SA
Chen et al. (2018)	√		√		GA
Lopes et al. (2019)	√		√	√	–
Şahin and Kellegöz (2019)	√		√	√	PSO
Bautista and Pereira (2007)		√	√		ACO
Chica et al. (2010)		√	√		ACO
Bautista and Pereira (2011)		√	√	√	–
Chica et al. (2011)		√	√		GA
Chica et al. (2012)		√	√		ACO
Rada-Vilela et al. (2013)		√			ACO
Chica et al. (2016)		√		√	GA
This research	√	√	√	√	ACS_LS

Table 2
Parameters of the TSALBP with multi-manned workstations.

i, h	Index of task
j	Index of workstation
k	Index of operator
I	Set of all the tasks
J	Set of all the workstations
K	Set of all the operators in each workstation
P^0	Set of tasks that have no immediate predecessors;
$P(i)$	Set of immediate predecessors of task i
n	The total number of tasks
u_{max}	The admitted maximum number of operators in each workstation
CT	Cycle time
A	The global available area
M	A very large positive number
t_i	The processing time of task i
a_i	The area information of task i

- We first formulate a new assembly line balancing problem with multi-manned and space constraints. A new mixed-integer linear mathematical model (MILP) is proposed.
- A memetic ant colony optimization (ACS_LS) is designed to tackle this new problem. This algorithm has a new solution generation, a new solution update mechanism, and a referenced local search to enhance its performance.

3. Multi-manned assembly line balancing with time and space constraints

In a typical multi-manned assembly line, all the tasks are needed to be assigned to a series of multi-manned workstations. There are u_{max} operators simultaneously performing all the different tasks of the production line. Each task i requires a processing time t_i and has a set of direct predecessors $P(i)$. Each task is only assigned to one station and its complete time in this station does not exceed the cycle time. For every pair of tasks (i, h) $(h \in P(i))$, the index of workstation allocated with task h should be less than or equal to that with task i. If they are assigned to the same workstation, the starting time of task i is not less than the finishing time of task h. Since u_{max} operators are allowed in one single station, each task should be assigned to exact one operator and the starting time and finishing time of the task also need to be determined when assigning tasks.

In TSALBP with multi-manned workstations, space constraint is also considered since the length of stations is limited and the required tools and components occupy the station space. Hence, apart from the time attribute, a task i is associated with a required area a_i. A global available area A is determined to limit the space of each station. The required space for each station is equal to the sum space of the tasks assigned to that station, and it is not more than the global available area.

The single-objective TSALBP with multi-manned workstations can be stated as: given the admitted maximum number of operators in each workstation, the cycle time, the global available area and a set of tasks with processing time and space information and precedence graph, all the tasks need to be assigned into the stations fulfilling the task assignment constraint, precedence relation constraint, cycle time constraint and space constraint. The details of the proposed model is presented in the following sub-sections.

Table 3
Decision variables of the TSALBP with multi-manned workstations.

W_{ih}	(binary) If task i and h are assigned to the same operator and task i is performed immediately before task h, *the value is* 1. 0, otherwise.
X_{ijk}	(binary) If task i is operated by operator k at workstation j, *its value is* 1. 0, otherwise.
Y_{jk}	(binary) If operator k is employed in workstation j, 1. 0, otherwise.
Z_j	(binary) If workstation j is opened, 1. 0, otherwise.
FT_i	(continuous) The finishing time of task i.

3.1. Assumptions and notations

The assumption of time and space assembly line balancing problem with multi-manned workstations are given as follows:

(1) Only one type product is assembled in the multi-manned assembly lines.
(2) The processing times of tasks and the precedence relationships between tasks are known deterministically.
(3) The travel times of operators are ignored.
(4) The cycle time is fixed during the multi-manned assembly lines.
(5) Parallel tasks and parallel workstations are not allowed.
(6) Each task must be assigned to only one workstation.
(7) The maximum number of operators in each workstation is determined.

Besides, the parameters and decision variables of the TSALBP with multi-manned workstations are presented in Table 2 and 3, respectively.

3.2. Minimization function and constraints

The model aims to minimize the number of operators and workstations to achieve the effectiveness of TSALBP with multi-manned workstations. As proposed in Fattahi et al. (2010), we optimize the number of operators as the primary objective and the number of multi-manned workstations. This objective function is defined in Equation (1).

$$\text{minimize} \sum_{j\in J}\sum_{k\in K} Y_{jk} + \frac{1}{u_{max}\times n+1}\sum_{j\in J} Z_j \tag{1}$$

In this objective function, weight coefficient $\frac{1}{u_{max}\times n+1}$ ensures that the number of multi-manned workstations is less than 1 and hence, the number of workstations has a lower priority rather than the number of operators.

The model is restricted to constraints defined from Equation (2) to Equation (19).

$$\sum_{j\in J}\sum_{k\in K} X_{ijk} = 1, \forall i \in I \tag{2}$$

$$\sum_{j\in J}\sum_{k\in K}(j \times X_{hjk}) - \sum_{j\in J}\sum_{k\in K}(j \times X_{ijk}) \le 0, \forall i \in I - P^0, h \in P(i) \tag{3}$$

$$FT_i \le CT, \forall i \in I \tag{4}$$

$$FT_i - FT_h + M \times \left(1 - \sum_{k\in K} X_{ijk}\right) + M \times \left(1 - \sum_{k\in K} X_{hjk}\right) \ge t_i, \forall i \in I - P^0, h \in P(i), j \in J \tag{5}$$

$$FT_h - FT_i + M \times \left(1 - X_{ijk}\right) + M \times \left(1 - X_{hjk}\right) + M \times (1 - W_{ih}) \ge t_h, \forall (i,h) | i \ne h \wedge i,h \in I, j \in J, k \in K \tag{6}$$

$$FT_i - FT_h + M \times \left(1 - X_{ijk}\right) + M \times \left(1 - X_{hjk}\right) + M \times W_{ih} \ge t_i, \forall (i,h) | i \ne h \wedge i,h \in I, j \in J, k \in K \tag{7}$$

$$\sum_{i\in I}\sum_{k\in K}(x_{ijk}\times a_i)\le A,\forall j\in J \tag{8}$$

$$\sum_{i\in \mathrm{I}}X_{ijk}\le n\times Y_{jk},\forall j\in J,k\in K \tag{9}$$

$$\sum_{i\in \mathrm{I}}X_{ijk}\ge Y_{jk},\forall j\in J,k\in K \tag{10}$$

$$Y_{j,k+1}\le Y_{jk},\forall j\in J,k=1,2,\cdots,u_{max}-1 \tag{11}$$

$$\sum_{i\in \mathrm{I}}\sum_{k\in \mathrm{K}}X_{ijk}\le(n\times u_{\max})\times Z_j,\forall j\in J \tag{12}$$

$$\sum_{i\in \mathrm{I}}\sum_{k\in \mathrm{K}}X_{ijk}\ge Z_j,\forall j\in J \tag{13}$$

$$Z_{j+1}\le Z_j,\forall j=1,2,\cdots,m-1 \tag{14}$$

$$FT_i\ge t_i,\forall i\in I \tag{15}$$

$$X_{ijk}\in\{0,1\},\forall i\in I,j\in J,k\in K \tag{16}$$

$$W_{ih}\in\{0,1\},\forall(i,h)|i\ne h\wedge i,h\in I,j\in J,k\in K \tag{17}$$

$$Y_{jk}\in\{0,1\},\forall j\in J,k\in K \tag{18}$$

$$Z_j\in\{0,1\},\forall j\in J \tag{19}$$

Task assignment constraint (2) is one of the essential constraints of TSALBP with multi-manned workstations. It ensures that each task is assigned exactly to one position of one operator at one workstation. Precedence constraint (3) correspond to that for each pair of tasks i and its immediate predecessor h, task h should be assigned before task i. Cycle time constraint (4) implies that each task must be finished within the cycle time. Sequencing constraints (5)–(7) control the sequence-dependent starting time of each task. For every pair of tasks i and h, if task h is the immediate predecessor of task i and they are assigned to the same workstation, constraint (5) becomes active and ensures that task i is started after finishing the task h. If tasks i and h have no precedence relation and task i is performed immediately before task h by the same operator, constraint (6) becomes $FT_h - FT_i \ge t_h$. Otherwise, constraint (7) becomes $FT_i - FT_h \ge t_i$. Space constraint (8) ensures that the required area of any workstation is not greater than the global available area. Variable constraints (9)–(19) determine the variables. Specifically, constraint (9) defines that if at least one task is assigned to the k-th operator of the j-th workstation, then Y_{jk} must be one. And constraint (11) further ensures that operators are loaded in an increasing manner. Constraints (12)–(14) confines that if at least one task is assigned to the j-th workstation, then Z_j must be one, and the workstations are also loaded in an increasing manner. Constraint (15) requires that the finishing time of each task should not smaller than the processing time of each task. Constraints (16)–(19) state the binary variables.

3.3. Lower bound

In this subsection, a lower bound of the number of operators and workstations is proposed. A convenient lower bound can be used as a termination condition and help to efficiently find the optimal solution obtained by the CPLEX solver in large-scale instances. When the objective value obtained by the algorithms reaches the lower bound, the algorithm will stop running. Therefore, the use of a lower bound is important and can save computational time. The objective function shown in equation (1) has two main components: the number of operators (NO) and the number of workstations (NW). The lower bounds for NW and NO can be calculated by the equations (20)–(21). Since two attributes including time and space are considered in our new problem, the lower bound should be defined in the above two aspects.

$$LB_{NW}=\max\left\{\left\lceil\frac{\sum_{i\in I}t_i}{u_{max}\times CT}\right\rceil,\left\lceil\frac{\sum_{i\in I}a_i}{A}\right\rceil\right\} \tag{20}$$

$$LB_{NO}=\max\left\{\left\lceil\frac{\sum_{i\in I}t_i}{CT}\right\rceil,\left\lceil\frac{\sum_{i\in I}a_i}{A}\right\rceil\right\} \tag{21}$$

In terms of time attribute of NW, taking into account the fact that $NW\times u_{max}\times CT$ must be greater than or equal to the sum of the processing time $\sum_{i\in I}t_i$, The lower bounds for NW obviously can be defined by $\left\lceil\frac{\sum_{i\in I}t_i}{u_{max}\times CT}\right\rceil$. In view of the space attribute, the required area of any workstation is not greater than the global available area A, which is the space constraint. That is, the required area of all workstations, which is equal to the total of the area information $\sum_{i\in I}a_i$, is not greater than $NW\times A$. Then, the lower bounds for NW is also defined by $\left\lceil\frac{\sum_{i\in I}a_i}{A}\right\rceil$. Thus, the simple lower bound for the number of workstations can be stated as the equation (20).

The lower bound of the number of operators also needs to be defined from the two aspects. Regarding the time attribute, since all the tasks should be processed by one operator before the cycle time, the required time $NO\times CT$ includes the actual sum of processing time $\sum_{i\in I}t_i$ and idle time. Hence, the low bound od NO can be calculated by $\left\lceil\frac{\sum_{i\in I}t_i}{CT}\right\rceil$. As far as the space attribute, since each workstation must be arranged at least one operator, the low bound of NO is equal to that of NW and is defined as $\left\lceil\frac{\sum_{i\in I}a_i}{A}\right\rceil$. Eventually, the simple lower bound for the number of operators is stated as the equation (21). And the low bound of the objective is determined as equation (22).

$$LB_S=LB_{NO}+\frac{LB_{NW}}{u_{max}\times n+1} \tag{22}$$

3.4. A numerical example

A numerical example is presented here to provide with a better insight of the new problem. Table 4 shows the precedence graph, processing time, and area information of this example. The maximum number of operators for each workstation is 3, the cycle time is 7, and the global available area is 14. This example is solved through the proposed mathematical model, coded by the GAMS/CPLEX solver. The optimal solution found by the method is shown in Fig. 1. Fig. 1(a) shows the tasks assigned to the operators and stations together with the starting time of the tasks. Fig. 1(b) presents the area of each station.

In the optimal solution, the total number of operators and stations are 4 and 2, respectively. And since more than one operator are allowed for each station, different tasks can be processed at the same time such as tasks 2 and 3. Specifically, the workload of stations 1 and 2 are equal to

Table 4
Details of the numerical example of Section 3.4.

Task i	Immediate predecessors $P(i)$	Processing time t_i	Area information a_i
1	0	1	3
2	1	3	2
3	1	2	4
4	1	2	2
5	2, 3	3	3
6	4	2	2
7	6	4	2
8	5	2	3
9	8	3	1

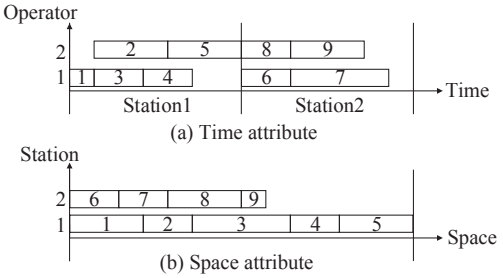

Fig. 1. The optimal solution for the numerical example of Section 3.4.

11, a higher time than the allowed cycle time of 7. If only one operator is assigned in each station, the workload cannot exceed the cycle time. Therefore, it can be concluded that less stations are needed to organize the required production due to the characteristics of multi-manned workstations. It should be noted that task 2 should be started after the completion of its predecessor task 1, even if it is the first task processed by operator 2 in the first station. Besides, the area information of the two stations are equal to 14 and 8, respectively. These values are lower than the global available area of 14. Please note that these space constraints also affect the task assignment.

4. Memetic ant colony optimization

Due to the NP-hard nature of TSALBP with multi-manned workstations, the difficulty of obtaining optimal solution increases exponentially as the scale of problem increases. Therefore, an ACO algorithm is proposed in this section to optimize the new problem. This algorithm is a population-based algorithm which mimics the process of food-search by ants in nature. Since the seminal ACO work by Dorigo et al. (1996), the method has been successfully applied to complex optimization problems, such as job shop and flow shop scheduling (Elmi & Topaloglu, 2017; Kurdi, 2018). There are several ACO variants such as ant system, ant colony system, max–min ant system, ranked-based ant system and best-worst ant system (Cordón García, Herrera Triguero, & Stützle, 2002).

4.1. ACS and local search integration

This paper uses a memetic version of the ant colony system (ACS) with a local search procedure to conveniently tackle the problem. From now on, we name it as the ACS_LS algorithm. This algorithm is a joint application of the global (i.e., the ACS optimization algorithm) and a local search. This approach is called memetic algorithm and its efficacy in solving real-world problems has been proved in previous works (Chica et al., 2012).

The ACS algorithm starts with an ant colony with a predetermined number. Each ant tries to assign tasks into workstations according to the pheromone and constraints related to TSALBP with multi-manned workstations in order to build a feasible solution. Sequentially, the global pheromone and the best solution are updated. The above procedures are repeated until the termination condition is reached. The outline of proposed ACS algorithm is shown in Fig. 2.

In our proposed algorithm, four parameters are predetermined, specifically including the population size (*PS*), parameters α and β determining the importance of pheromone and heuristic information, and parameter q_0. And two matrix variables $\Delta\tau^s_{ur}$ and τ_{ur} are used here to represent the pheromone quantity. The former refers to local pheromone which is utilized in the solution generation phase. Whenever an ant i generated a new solution, the $\Delta\tau^s_{ur}$ is equal to 1 if the task u is selected in the r-th iteration by the ant s. And the latter τ_{ur} is named global pheromone and used to help every ant to select task in each iteration. It represents the pheromone of task u in *r-th* iteration and is updated in the pheromone release phase. After all the ants complete building a new solution for the problem, the best solution is updated by the new generated solutions.

Besides, a local search procedure is used in conjunction to the ACS algorithm to create a memetic ACS algorithm (ACS_LS) for tackling the multi-manned assembly line balancing with time and space constraints. The referenced local search refinement is applied to the best solution during each iteration to improve its quality. The details of the solution generation, pheromone release, the best solution updating phases and local search procedure are explained in the next sub-sections.

4.2. Solution generation

In the proposed ACS_LS algorithm, each ant builds a feasible solution in a way that task assignment constraint, precedence constraint, cycle time constraint, sequencing constraints and space constraint should be satisfied. Before building a feasible solution by any ant i, the current workstation wc is set 0 and the iteration counter r is set 1. This procedure has seven steps which are detailed below.

Step 1: Open a new multi-manned workstation.

Open a new multi-manned workstation ($wc = wc + 1$) with one operator ($ws = 1$).

Step 2: Construct the candidate set of available tasks.

When a new multi-manned workstation is set, the candidate set of available tasks C^s_r should be determined. A task is available for this set based on the following criteria:

(1) It has not been assigned (task assignment constraint).
(2) Its predecessors have been assigned (precedence constraint).
(3) L operators ($L \leq ws$ and $L \neq 0$) can process the task (cycle time constraint).
(4) The task does not violate the space constraints of the current workstation (space constraint).

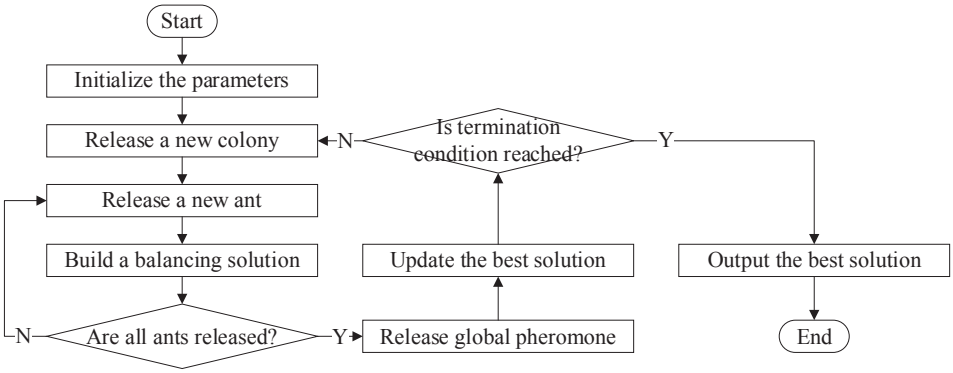

Fig. 2. The outline of ACS algorithm.

Table 5
The heuristic information used by ACS_LS.

No.	Heuristic information
1	The label of task
2	The processing time of task
3	The space information of task
4	The number of all predecessors
5	The number of all successors
6	The total processing time of all predecessors
7	The total processing time of all successors
8	The total space information of all predecessors
9	The total space information of all successors
10	The number of all immediate predecessors
11	The number of all immediate successors
12	The total processing time of all immediate predecessors
13	The total processing time of all immediate successors
14	The total space information of all immediate predecessors
15	The total space information of all immediate successors
16	Random heuristic for the above

If the candidate set C_r^s is empty, which means the current workstation is full, the procedure returns to step6; otherwise, it goes on the next step.

Step 3: Select a task.

A task is selected from the candidate set to current workstation according the ant colony system state transition rule proposed by Dorigo and Gambardella (1997) and Simaria and Vilarinho (2009). This rule considers the global pheromone variable in *r*-th iteration and the heuristic information. And 16 heuristics related to the TSALBP with multi-manned workstations are used here and provided in Table 5. The rule determines a task using equation (23).

$$i = \begin{cases} \underset{u \in C_r^s}{\operatorname{argmax}} \{[\tau_{ur}]^{\alpha} \cdot [\eta_u]^{\beta}\} if q \le q_0 \\ \underset{u \in C_r^s}{\operatorname{argmax}} Pr(u, r) if q > q_0 \end{cases} \tag{23}$$

where α, β and τ_{ur} are introduced above; q is a random generated number between (Scholl & Becker, 2006) and q_0 is predefined parameter determined by user; η_u is the heuristic information of task u; $Pr(u, r)$ is the selection probability of task u in the *r-th* iteration, which can be calculated using equation (24).

$$Pr(u, r) = \frac{[\tau_{ur}]^{\alpha} \cdot [\eta_u]^{\beta}}{\sum_{h \in C_r^s} [\tau_{hr}]^{\alpha} \cdot [\eta_h]^{\beta}} \tag{24}$$

According to this rule, there are two behaviors to select the task from the candidate set:

(1) If $q \le q_0$, the algorithm tends to select a task with the maximum value of joint global pheromone and heuristic information;
(2) If $q > q_0$, the algorithm determines a task based on the probability, which may make the algorithm choose a non-explored path.

After a task u is selected in the *r-th* iteration by ant s, the value of local pheromone $\Delta\tau_{ur}^s$ is updated and equal to 1. Then, the iteration counter r pluses 1.

Step 4: Determine the set of available operators

Since more than one operator is allowed in the multi-manned workstation, the ant should assign the selected task to a specific operator. Hence, the set of available operators needs to be determined. An operator k is available if the start time of the selected task u (ST_{uk}) in this operator plus its processing time cannot exceed the cycle time and the label of this operator does not exceed ws (cycle time constraint, precedence constraint and sequencing constraints).

Step 5: Select an operator

The MALBP involves two types idle times: sequence-dependent idle times and remaining idle times. In order to reduce the idle times, the following rules are employed to select the operator from the set of available operators. The rules are:

(1) Select an operator with the minimum value of ST_{uk};
(2) Tie broken, select an operator with the minimum sequencing idle time;
(3) Tie broken, randomly select an operator.

After finishing the selection of operator, return to Step2.

Step 6: Calculate the efficiency and select the optimal configuration

If ws is smaller than u_{max}, the task assignment plan of workstation wc with ws operators is set as $Configuration_{ws}$. Then we set $ws = ws + 1$ and the procedure jumps to Step2 to reassign the tasks. If not, the configuration with maximum efficiency is selected as the current workstation, and r is equal to the number of assigned tasks plus 1. Note that the efficiency of the configuration with ws operators E_{ws} is equal to $\frac{WT_{wc}}{ws \cdot CT}$ where WT_{wc} is the total processing time in current station wc.

Step 7: Evaluate the termination condition and calculate the objective function.

If there are tasks not assigned, the procedure returns to Step1. Otherwise, the algorithm calculates the objective function and hence complete the construction of a feasible solution.

To sum up, we show the pseudocode of the solution generation procedure in Algorithm 1.

Algorithm 1. (*Solution generation procedure*)

```
wc = 1; s = 1;
for (all tasks are not assigned) do
  for (ws = 1 to umax) do
    r = s;
    Construct the candidate set of available tasks according to criteria in step 2;
    if the candidate set is not empty then
      Select a task u in r-th iteration according to transition rule (equation (23));
      Update local pheromone Δτ_ur^s;
      Determine the set of available operators for task u;
      Select an operator based on the rules in step 5 to perform task u;
      r = r + 1;
      Update the candidate set of available tasks;
    end if
    Calculate the efficiency of wc-th station with ws operators;
    It_ws = r; (the current iteration of wc-th station with ws operators)
  end for
  Select the configuration with the maximum efficiency for wc-th station;
  Assume the selected configuration contains m operators;
  s = It_m; wc = wc + 1;
end for
return the complete solution.
```

4.3. Pheromone release

After all the ants build solutions, a pheromone release strategy proposed by Baykasoglu and Dereli (2008) is employed to update the global pheromone quantity. This strategy contains two main operations: deposition and evaporation. The deposition aims to perform the trails of all ants and updates the pheromone using the equation (25).

$$\tau_{ur} = \tau_{ur} + \Delta\tau_{ur}^s \tag{25}$$

In the deposition, the trails of some ants who build the solutions with worse performance are also stored by using the equation (25), which will send bad messages to the ants in the next iteration and further induce them to explore bad paths. Hence, the evaporation tries to release these trails in order to avoid this situation. It first calculates the average number of workstations (*anw*) and average number of operators (*ano*) in current colony. And then, if the number of workstations and number of operators in the solution obtained by ant s are larger than or equal to *anw* and *ano* respectively, the global pheromone trails are evaporated by

equation (26).

$$\tau_{ur} = \tau_{ur} - \Delta\tau_{ur}^{s} \tag{26}$$

4.4. The best solution updating

When a new solution is generated, the best solution needs to be updated. The best solution can be replaced by the new generated solution based the two following criteria is met:

(1) The objective function of the new solution is lower than the one of the best solution.
(2) The objective functions of them are equal and the variant workload (VW) of the new solution is lower than the one of the best solution.

The first criterion makes sure that the total number of operators and workstations are minimized as the primary and secondary objectives respectively to achieve the effectiveness of TSALBP with multi-manned workstations. If the first criterion does not determine the best solution, the second criterion works to promise that the differences between operators' workload and that between workstations' space are as small as possible without increasing the number of operators and workstations. The variant workload (*VW)* is defined in Equation (27).

$$VW = \frac{\sqrt{\sum_{j\in J}\sum_{k\in K}\left\{Y_{jk} \times \left[\sum_{i\in I}\left(X_{ijk} \times t_i\right) - t_{avg\_op}\right]^2\right\}}}{\sum_{j\in J}\sum_{k\in K} Y_{jk}} \tag{27}$$

$$t_{avg\_op} = \frac{\sum_{i\in I} t_i}{\sum_{j\in J}\sum_{k\in K} Y_{jk}} \tag{28}$$

where t_{avg_op} is the average processing time per operator and $\sum_{j\in J}\sum_{k\in K} Y_{jk}$ is the total number of operators.

4.5. Local search method of the memetic algorithm

The local search procedure used in our memetic algorithm aims to improve the best solution by insert operator defined on the search space. Since the referenced local search has been successfully applied in the flow shop scheduling (Pan, Gao, Wang, Liang, & Li, 2019), the proposed algorithm employed it to explore the search space around the best solution. It works on the task assignment sequence which is determined in the solution generation. In the procedure, the task is selected in order and inserted into all positions meeting the precedence relations to generate a set of partial solutions. If the new generated solution with best objective function is better than the current best solution, the current best solution is replaced with the new generated solution. The above process is repeated until there is no improvement found for *NA* times (i.e., the upper limit for the local search procedure). Note that the tasks are selected in order given by the task assignment sequence of the best solution before executing referenced local search. The description of the method is shown in Algorithm 2.

Algorithm 2. (*Referenced local search procedure*)

```
1: input: a best solution X;
2: output: a best solution X;
3: begin:
5:   counter = 0; i = 1;
6:   while counter < NA do
7:     extract task λ from the referenced task assignment sequence;
8:     generate a set of partial solutions by reinsert λ into the all possible positions;
9:     X'= the best solution in the set of partial solutions;
10:    if f(X') < f(X) then
11:      X = X'; counter = 0;
12:    else
13:      counter = counter + 1;
14:    end if
15:    i=(i + 1) % N;
16:  end while
18: return X;
```

(continued on next column)

(continued)

5. Experimental study

This section presents and analyzes the designed three experiments of this work to test the performance of the proposed model and ACO algorithm. First, the experimental setup and benchmark instances are introduced in Section 5.1. The parameters of the proposed memetic ant colony optimization algorithm ACS_LS are specified in Section 5.2. The three main experiments of the study are analyzed in Sections 5.3, 5.4, and 5.5. We first compare the proposed algorithm with respect to the results by GAMS/CPLEX on the lower bound in Section 5.3. Then, the heuristic rules embedded into the ACS_LS algorithm are compared in Section 5.4. Finally, we test the effectiveness of the proposed local search in Section 5.5.

5.1. Experimental setup and problem instances

This paper has carried out three sets of experiments to measure the optimization of the proposed mathematical model and ACS_LS algorithm. First, a calibration experiment is conducted to find the best parameter combination for the ACS_LS algorithm. Second, this paper compares the ACS_LS with the optimal solutions obtained by General Algebraic Modeling System 23.0 (GAMS)/CPLEX to evaluate the proposed model and ACS_LS algorithm in the same instances. Then, the ACS algorithm embedded with different heuristic rules compares with each other in all instances to find a best heuristic rule. Finally, the effectiveness of the local search is tested by comparing the ACS_LS with ACS. The ACS and ACS_LS algorithms are coded in the C++ programming language and are run on a computer with Intel^{R} Core^{TM} i7 4790 processor running at 3.20 GHz with 8GBytes of RAM. In addition, the relative percentage deviation (RPD) is applied to measure the results of all experiments. The RPD can be calculated by the following equation (29) where $Objective_{sol}$ is the objective function of each solution and $Objective_{best}$ means the best objective function obtained by all algorithms.

$$RPD = \frac{Objective_{sol} - Objective_{best}}{Objective_{best}} \times 100 \tag{29}$$

Besides, a set of benchmarks consisting of 154 instances, which are drive from 10 calibration instances and 144 comparison instances, are first generated. Among them, 10 calibration instances shown in Table 6 are used to determine the parameters of ACS_LS and 144 comparison instances shown in Table 7 are applied to evaluate the performance of proposed model and algorithm. The task times and precedence relation of these instances can be downloaded from the assembly line website (http://assembly-line-balancing.mansci.de). In order to add spatial

Table 6
The calibration instances for TSALBP with multi-manned workstations.

Instances	n	t_{min}	t_{max}	t_{sum}	Cycle time
Heskia28	28	1	108	1024	271
Lutz32	32	100	1400	14140	1899
Kilbridge45	45	3	55	552	142
Warnecke58	58	7	53	1548	100
Weemag75	75	2	27	1499	66
Lutz89	89	1	74	1644	122
Mukherje94	94	8	171	4208	244
Arcus111	111	10	5689	150399	11111
Barthol148	148	3	83	4243	140
Scholl297	297	5	1386	69655	2532

Table 7
The instances for TSALBP with multi-manned workstations.

Instances	n	t_{min}	t_{max}	t_{sum}	Cycle time used in this paper
Merten7	7	1	6	29	6, 7, 8, 10, 15, 18
Bowman8	8	3	17	75	17, 21, 24, 28, 31
Jaeschke9	9	1	6	37	6, 7, 8, 10, 18
Jackson11	11	1	7	46	7, 9, 10, 13, 14, 21
Mansoor11	11	2	45	185	45, 54, 63, 72, 81
Mitchell21	21	1	13	105	14, 15, 21, 26, 35, 39
Roszieg25	25	1	13	125	14, 16, 18, 21, 25, 32
Heskia28	28	1	108	1024	138, 205, 216, 256, 324, 342
Buxey29	29	1	25	324	27, 30, 33, 36, 41, 47, 54
Sawyer30	30	1	25	324	25, 27, 30, 36, 41, 54, 75
Lutz32	32	100	1400	14,140	1414, 1572, 1768, 2020, 2357, 2828
Gunther35	35	1	40	483	41, 44, 49, 54, 61, 69, 81
Kilbridge45	45	3	55	552	57, 79, 92, 110, 138, 184
Hahn53	53	40	1775	14,026	2004, 2338, 2806, 3507, 4676
Warnecke58	58	7	53	1548	54, 56, 65, 71, 82, 92, 111
Tonge70	70	1	156	3510	176, 364, 410, 468, 527
Weemag75	75	2	27	1499	28, 32, 35, 38, 40, 47, 56
Arcus83	83	233	3691	75,707	5048, 5853, 6842, 7571, 8412, 8998, 10,816
Lutz89	89	1	74	1644	75, 83, 92, 110, 118, 137, 150
Mukherje94	94	8	171	4208	176, 183, 192, 201, 211, 324, 351
Arcus111	111	10	5689	150,399	5755, 8847, 10027, 10743, 11378, 17,067
Barthol148	148	3	83	4243	84, 99, 101, 115, 118, 157, 170
Scholl297	297	5	1386	69,655	1394, 1483, 1659, 1834, 2049, 2322, 2488, 2787

information to the instances, we follow the procedure presented in Chica et al. (2010), where the global available area is equal to the predefined cycle time and the spatial information of each task i is equal to the temporal information of task n-i + 1 (i.e., $a_i = t_{n-i+1}$).

5.2. Setting the ACS_LS parameters

Since the parameters have great influence on the performance of proposed ACS_LS algorithm, the full factorial design and Analysis of Variance (ANOVA) technique is employed to select the best parameter configuration. The ANOVA is an important parametric statistical inference tool used to check the three main hypotheses: normality, homoscedasticity and independence of the residuals. For the ACS_LS algorithm, four parameters need to be calibrated: (1) The population size *PS* at 7 levels: 40, 80, 120, 160, 200, 240 and 280; (2) Parameter α at 8 levels: 0.05, 0.15, 0.25, 0.5, 0.75, 0.8, 0.85 and 0.9; (3) Parameter β at 3 levels: 1, 2 and 3; (4) Parameters q_0 at 5 levels: 0.15, 0.25, 0.35, 0.5 and 0.85. Through the full factorial design, there are a total of $7 \times 8 \times 3 \times 5 = 840$ experiment configurations and all these configurations are run with all 10 calibration instances. Each instance is solved 10 times under the stopping criteria, a CPU time limit of $n \times n \times 5$ ms, and the minimum solution is selected to calculate the RPD of each parameter configuration as the final response value. The Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all the factors in the ANOVA calibration experiment for the proposed ACS_LS are depicted in Fig. 3 and the ANOVA results for the parameters are shown in Table 8.

Table 8
ANOVA results for the parameters.

Sources	df	Type III sum of squares	Mean square	*F*-ratio	*P*-value
Corrected Model	839	2.3128	0.002757	0.61	1.000
PS	6	0.6116	0.101933	22.47	0.000
α	7	0.0178	0.002548	0.56	0.788
β	2	0.0071	0.003565	0.79	0.456
q_0	4	0.7556	0.188893	41.65	0.000
$PS*\alpha$	42	0.0682	0.001624	0.36	1.000
$PS*\beta$	12	0.0115	0.000958	0.21	0.998
$PS*q_0$	24	0.0444	0.001851	0.41	0.995
$\alpha*\beta$	14	0.0147	0.001047	0.23	0.999
$\alpha*q_0$	28	0.0242	0.000863	0.19	1.000
$\beta*q_0$	8	0.0088	0.001099	0.24	0.983
$PS*\alpha*\beta$	84	0.0980	0.001167	0.26	1.000
$PS*\alpha*q_0$	168	0.1896	0.001128	0.25	1.000
$PS*\beta*q_0$	48	0.0517	0.001078	0.24	1.000
$\alpha*\beta*q_0$	56	0.0647	0.001155	0.25	1.000
$PS*\alpha*\beta*q_0$	336	0.3450	0.001027	0.23	1.000
Error	7560	34.2899	0.004536		
Total	8399	36.6027			

In the ANOVA results, the smaller the *P*-value of the factor, the great the effect of this factor on the ACS_LS algorithm. It is observed from Table 8 that the *P*-values of the levels of *PS* and q_0 are equal to 0.00 which means they have the greatest influence on the ACS_LS algorithm. Apart from these two parameters, the levels of β have the smallest *P*-values with 0.456. Besides, the levels of other parameters and the interaction between the any number of four parameters have the similar *P*-value close to 1.000. Since there are many close *P*-values, it is difficult to judge the effect of each parameter and interaction. In this situation, the *F*-ratio can be employed to make further determination. the larger the *F*-ratio, the great the effect of the factor on the ACS_LS algorithm. For

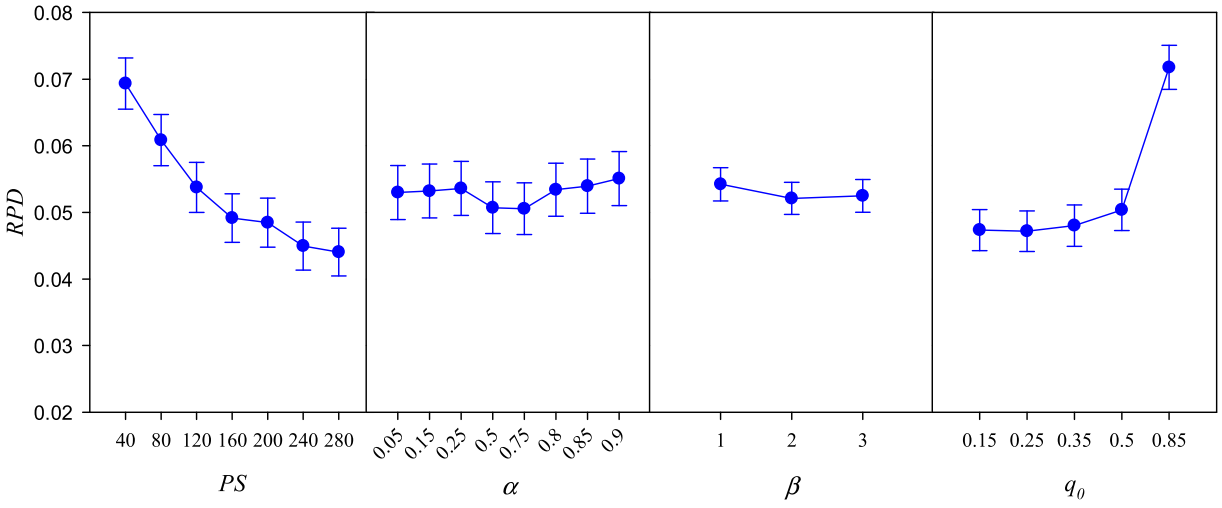

Fig. 3. The Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all the factors in the ANOVA calibration experiment for the proposed ACS_LS.

parameters PS and q_0 with the same P-value, the F-ratio of q_0 is 41.65 larger than that of PS, and hence the parameter q_0 has the greatest impact on the performance of ACS_LS algorithm. Later, the AVOVA determines the best parameter configuration and shows the results in Fig. 3. Clearly, the parameters α of 0.75, β of 2 and q_0 of 0.25 result in a most efficient ACS_LS. As for the population size PS, the performance of ACS_LS gets better with the increase of the value of PS, but the magnitude gets increasingly smaller. The difference between RPD values of 240 and 280 is less than 0.001, and hence PS of 280 is selected as the best level. From the analysis we conclude that the best parameter configuration is $PS = 280$, $\alpha = 0.75$, $\beta = 2$ and $q_0 = 0.25$.

5.3. Comparison with lower bounds for small-scaled instances

A set of comparison experiments has been performed by solving 38 small-scaled instances using the proposed MILP model coded in GAMS/CPLEX and the ACS_LS algorithm. The running time of GAMS/CPLEX is controlled within 1 h, and ACS_LS algorithm is solved under the stopping criteria, a CPU time limit of $n \times n \times 5$ ms.

This section first compares GAMS/CPLEX and ACS_LS with each other to evaluate the performance of proposed model and ACS_LS algorithm in small-scaled instances. For the GAMS/CPLEX solver, we first use the default emphasis parameter 0 (balancing the optimality and feasibility) to solve each instance. If an optimal solution is found, we continue use the emphasis parameter 2 (emphasize optimality over feasibility) to further solve this instance; otherwise, we use the emphasis parameter 1 (emphasize feasibility over optimality) to find a feasible solution. The best results and corresponding parameters are reported.

The comparison results of GAMS/CPLEX and ACS_LS are shown in Table 9. As seen, 16 instances obtain the optimal solution in small-scaled instances via GAMS/CPLEX within 1 h. And in these 16 instances, ACS_LS algorithm also gets the same result within shorter time. In the instances of Mitchell21 when CT is 14 or 15, Roszieg25 when CT is 18, Heskia28 when CT is 205 or 216, GAMS/CPLEX just outputs a solution but not the optimal solution; while ACS_LS algorithm can obtain significantly better results. In other instances, GAMS/CPLEX can hardly obtain a feasible solution within 1 h. These comparison results conclude that GAMS/CPLEX has great performance when the number of tasks is less than 21, and its performance grows worse with the increasing of the instance size. Nevertheless, the proposed ACS_LS algorithm is an effective way for solving this new problem.

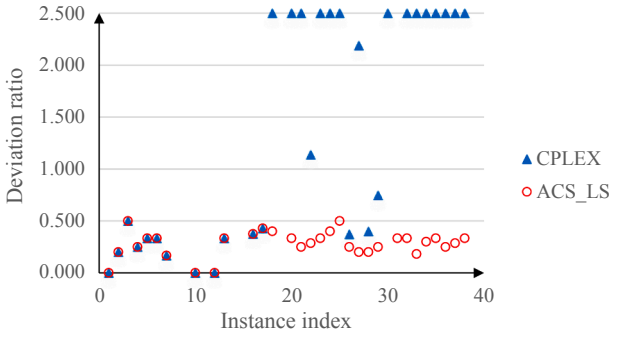

Fig. 4. Deviation ratio of the CPLEX/ACS_LS.

Table 9
Comparison results of GAMS/CPLEX and ACS_LS.

No.	Instances	*n*	*CT/A*	CPLEX solver Emphasis values	Results	Lower bound	Gap	CPU	ACS_LS Results	CPU
1	Merten7	7	18	0, 2	**2.091**	2.091	0.00	0.446	**2.091**	0.493
2	Bowman8	8	17	0, 2	**6.240**	6.240	0.00	0.384	**6.240**	0.509
3	Bowman8	8	21	0, 2	**6.240**	6.240	0.00	0.740	**6.240**	0.617
4	Bowman8	8	24	0, 2	**5.200**	5.200	0.00	0.838	**5.200**	0.539
5	Bowman8	8	28	0, 2	**4.160**	4.160	0.00	3.721	**4.160**	0.575
6	Bowman8	8	31	0, 2	**4.160**	4.160	0.00	3.048	**4.160**	0.564
7	Jaeschke9	9	7	0, 2	**7.250**	7.250	0.00	0.443	**7.250**	0.626
8	Jaeschke9	9	18	0, 2	**3.107**	3.107	0.00	1.461	**3.107**	0.589
9	Jackson11	11	9	0, 2	**6.176**	6.176	0.00	2.256	**6.176**	0.623
10	Jackson11	11	14	0, 2	**4.118**	4.118	0.00	5.167	**4.118**	0.611
11	Jackson11	11	21	0, 2	**3.088**	3.088	0.00	4.699	**3.088**	0.586
12	Mansoor11	11	54	0, 2	**4.118**	4.118	0.00	6.082	**4.118**	0.630
13	Mansoor11	11	63	0, 2	**4.118**	4.118	0.00	7.003	**4.118**	0.628
14	Mansoor11	11	72	0, 2	**3.088**	3.088	0.00	4.015	**3.088**	0.638
15	Mansoor11	11	81	0, 2	**3.088**	3.088	0.00	2.926	**3.088**	0.632
16	Mitchell21	21	14	0	**11.172**	9.140	0.18	3600.000	**11.172**	0.992
17	Mitchell21	21	15	0	**10.156**	9.141	0.10	3600.000	**10.156**	1.051
18	Mitchell21	21	21	0	13.140	5.078	0.61	3600.000	**7.109**	0.930
19	Mitchell21	21	26	0, 2	**5.078**	5.078	0.00	3147.802	**5.078**	0.877
20	Roszieg25	25	14	0, 1	–			3600.000	**12.158**	1.255
21	Roszieg25	25	16	0, 1	–			3600.000	**10.132**	1.122
22	Roszieg25	25	18	0	15.158	1.319	0.91	3600.000	**9.118**	1.459
23	Roszieg25	25	21	0, 1	–			3600.000	**8.105**	1.248
24	Roszieg25	25	25	0, 1	–			3600.000	**7.092**	1.423
25	Roszieg25	25	32	0, 1	–			3600.000	**6.079**	1.269
26	Heskia28	28	138	1	11.106	6.072	0.45	3600.000	**10.118**	1.705
27	Heskia28	28	205	0	16.129	1.083	0.93	3600.000	**6.071**	1.821
28	Heskia28	28	216	0	7.082	3.405	0.52	3600.000	**6.071**	1.653
29	Heskia28	28	256	1	7.071	2.616	0.63	3600.000	**5.059**	1.666
30	Heskia28	28	324	0, 1	–			3600.000	**4.047**	1.566
31	Heskia28	28	342	1	11.047	1.012	0.91	3600.000	**4.047**	1.489
32	Buxey29	29	27	0, 1	–			3600.000	**16.182**	1.474
33	Buxey29	29	30	0, 1	–			3600.000	**13.148**	1.570
34	Buxey29	29	33	0, 1	–			3600.000	**13.148**	1.613
35	Buxey29	29	36	0, 1	–			3600.000	**12.136**	1.387
36	Buxey29	29	41	0, 1	–			3600.000	**10.114**	1.503
37	Buxey29	29	47	0, 1	–			3600.000	**9.102**	1.718
38	Buxey29	29	54	0, 1	–			3600.000	**8.091**	1.514

Moreover, this section further compares GAMS/CPLEX and ACS_LS with the low bound calculated in equations (20–22). The deviation ratio (DR) is employed here to demonstrate the performance. It is equal to *RPD* over 100 where $Objective_{sol}$ is the objective function obtained by CPLEX or ACS_LS and $Objective_{best}$ is the low bound. The deviation ratio of the CPLEX/ACS_LS and lower bound is depicted in Fig. 4.

In Fig. 4, since GAMS/CPLEX does not get a feasible solution in some instances resulting in invalid ratio, a uniform ratio 2.5 is assumed to be its deviation ratio. Fig. 4 shows that three instances in GAMS/CPLEX and ACS_LS reach to the low bound and the deviation points of the optimal solution of and low bound in the 16 instances are in the 0–0.5 range. This means that the gap of the real optimal objective and proposed low bound still exits, and the ratio is about 0.5. Besides, all the deviation points of the ACS_LS and low bound are in the 0–0.5 range, which indicates that ACS_LS algorithm owns better performance in small-scaled instances.

5.4. Comparison of the heuristic rules

In order to test the performance of proposed ACS algorithm in middle-scaled and large-scaled instances, this section designs the comparison experiments between different ACS variants resulting from the use of the 16 heuristic rules defined in Table 5 (i.e. ACS1 using heuristic rule 1, ACS2 using heuristic rule 2, ACS3 using heuristic rule 3, etc.). For a fair comparison, all the ACS variants are solved under the same stopping criteria, a CPU time limit of $n \times n \times \rho$ milliseconds (in this section two values of ρ are considered: 10 and 20). Each algorithm is run on each one of the 144 comparison instances with two stopping criteria 10 times. A total of 46,080 experiments are collected and analyzed. For the lack of space, Tables 10–12 report the *RPD* of minimum and average objective function of algorithms ACS1, ACS3, ACS5, ACS7, ACS9, ACS11, ACS13 and ACS16 for small-, middle- and large-scaled instances respectively under a CPU time limit of $n \times n \times 10\ ms$.

Table 10
Comparison results of ACS variants for small-scaled instances.

Instances	*CT*	ACS1		ACS3		ACS5		ACS7		ACS9		ACS11		ACS13		ACS16	
		Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
Merten7	6	**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	**0.00**	0.00
	7	**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	**0.00**	0.00
	8	**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	**0.00**	0.00
	10	**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	**0.00**	0.00
	15	**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	**0.00**	0.00
	18	**0.00**	0.00	**0.00**	20.00	**0.00**	5.00	**0.00**	15.00	**0.00**	15.00	**0.00**	25.00	**0.00**	40.00	**0.00**	10.00
Bowman8	17	**0.00**	0.00	**0.00**	3.33	**0.00**	1.67	**0.00**	0.00	**0.00**	1.67	**0.00**	1.67	**0.00**	1.67	**0.00**	0.00
	21	**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	**0.00**	0.00
	24	**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	**0.00**	0.00
	28	**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	**0.00**	0.00
	31	**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	**0.00**	0.00
Jaeschke9	6	**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	**0.00**	0.00
	7	**0.00**	1.43	**0.00**	2.86	**0.00**	2.86	**0.00**	0.00	**0.00**	0.00	**0.00**	1.43	**0.00**	0.00	**0.00**	2.86
	8	**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	**0.00**	0.00
	10	**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	**0.00**	0.00
	18	**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	**0.00**	0.00
Jackson11	7	**0.00**	8.75	**0.00**	5.00	**0.00**	6.25	**0.00**	5.00	**0.00**	10.00	**0.00**	8.75	**0.00**	7.50	**0.00**	8.75
	9	**0.00**	3.33	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	1.67	**0.00**	1.67	**0.00**	0.00
	10	**0.00**	0.00	**0.00**	1.67	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
	13	**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	**0.00**	0.00
	14	**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	**0.00**	0.00
	21	**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	**0.00**	0.00
Mansoor11	45	**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	**0.00**	0.00
	54	**0.00**	15.00	**0.00**	20.00	**0.00**	17.50	**0.00**	17.50	**0.00**	12.50	**0.00**	17.50	**0.00**	22.50	**0.00**	20.00
	63	**0.00**	5.00	**0.00**	7.50	**0.00**	10.00	**0.00**	5.00	**0.00**	7.50	**0.00**	12.50	**0.00**	7.50	**0.00**	12.50
	72	**0.00**	3.33	**0.00**	13.33	**0.00**	6.67	**0.00**	3.33	**0.00**	10.00	**0.00**	10.00	**0.00**	6.67	**0.00**	10.00
	81	**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	**0.00**	0.00
Mitchell21	14	**0.00**	4.00	**0.00**	5.00	**0.00**	4.00	**0.00**	9.00	**0.00**	2.00	**0.00**	8.00	**0.00**	5.00	**0.00**	8.00
	15	**0.00**	10.00	**0.00**	7.78	**0.00**	7.78	**0.00**	10.00	**0.00**	8.89	**0.00**	8.89	**0.00**	6.67	**0.00**	8.89
	21	**0.00**	3.33	**0.00**	8.33	**0.00**	1.67	**0.00**	6.67	**0.00**	6.67	**0.00**	1.67	**0.00**	6.67	**0.00**	1.67
	26	**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	**0.00**	0.00
	35	**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	**0.00**	0.00
	39	**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	**0.00**	0.00
Roszieg25	14	**0.00**	0.00	**0.00**	0.83	**0.00**	0.00	**0.00**	0.00	**0.00**	0.83	**0.00**	0.83	**0.00**	0.00	**0.00**	0.00
	16	**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	**0.00**	0.00
	18	**0.00**	1.11	**0.00**	0.00	**0.00**	1.11	**0.00**	0.00	**0.00**	1.11	**0.00**	0.00	**0.00**	0.00	**0.00**	1.11
	21	**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	**0.00**	0.00
	25	**0.00**	11.67	**0.00**	13.33	**0.00**	10.00	**0.00**	8.33	**0.00**	13.33	**0.00**	11.67	**0.00**	6.67	**0.00**	3.33
	32	**0.00**	2.00	**0.00**	0.00	**0.00**	4.00	**0.00**	2.00	**0.00**	0.00	**0.00**	4.00	**0.00**	2.00	**0.00**	4.00
Heskia28	138	12.50	17.50	12.50	13.75	**0.00**	11.25	12.50	15.00	12.50	13.75	**0.00**	12.50	12.50	12.50	12.50	12.50
	205	**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	**0.00**	0.00
	216	20.00	20.00	**0.00**	18.00	20.00	20.00	**0.00**	16.00	20.00	20.00	**0.00**	18.00	20.00	20.00	20.00	20.00
	256	**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	**0.00**	0.00
	324	**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	**0.00**	0.00
	342	**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	**0.00**	0.00
Buxey29	27	**0.00**	5.33	**0.00**	4.67	**0.00**	3.33	**0.00**	4.67	**0.00**	4.67	**0.00**	5.33	**0.00**	5.33	**0.00**	2.67
	30	**0.00**	0.77	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.77	**0.00**	0.00
	33	**0.00**	6.67	**0.00**	7.50	**0.00**	3.33	**0.00**	6.67	**0.00**	5.83	**0.00**	6.67	**0.00**	7.50	**0.00**	5.00
	36	**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	**0.00**	0.00
	41	**0.00**	8.89	**0.00**	6.67	**0.00**	4.44	**0.00**	4.44	**0.00**	5.56	**0.00**	3.33	**0.00**	5.56	**0.00**	6.67
	47	**0.00**	8.75	**0.00**	11.25	**0.00**	7.50	**0.00**	6.25	**0.00**	7.50	**0.00**	8.75	**0.00**	10.00	**0.00**	7.50
	54	**0.00**	1.43	**0.00**	8.57	**0.00**	8.57	**0.00**	4.29	**0.00**	2.86	**0.00**	7.14	**0.00**	1.43	**0.00**	1.43

Table 11
Comparison results of ACS variants for middle-scaled instances.

Instances	*CT*	ACS1		ACS3		ACS5		ACS7		ACS9		ACS11		ACS13		ACS16	
		Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
Sawyer30	25	**0.00**	5.29	5.88	9.41	5.88	8.24	**0.00**	7.06	**0.00**	7.65	5.88	7.65	**0.00**	6.47	**0.00**	4.71
	27	6.67	12.67	6.67	14.67	6.67	12.67	6.67	12.00	6.67	13.33	6.67	14.00	**0.00**	10.67	6.67	10.67
	30	**0.00**	6.43	**0.00**	5.71	**0.00**	6.43	7.14	7.86	7.14	7.14	7.14	7.86	7.14	7.86	**0.00**	6.43
	36	**0.00**	8.18	9.09	9.09	9.09	9.09	**0.00**	8.18	9.09	9.09	9.09	9.09	9.09	9.09	**0.00**	8.18
	41	**0.00**	6.00	10.00	10.00	**0.00**	8.00	**0.00**	9.00	**0.00**	9.00	10.00	10.00	**0.00**	7.00	10.00	10.00
	54	14.29	14.29	14.29	14.29	**0.00**	12.86	**0.00**	11.43	**0.00**	11.43	**0.00**	12.86	**0.00**	12.86	**0.00**	8.57
	75	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
Lutz32	1414	**0.00**	5.38	**0.00**	5.38	**0.00**	4.62	**0.00**	4.62	**0.00**	3.85	**0.00**	3.85	**0.00**	3.08	**0.00**	3.08
	1572	**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	**0.00**	0.00
	1768	10.00	10.00	10.00	10.00	**0.00**	9.00	**0.00**	8.00	**0.00**	9.00	**0.00**	9.00	**0.00**	9.00	**0.00**	9.00
	2020	**0.00**	0.00	**0.00**	4.44	**0.00**	0.00	**0.00**	0.00	**0.00**	2.22	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
	2357	**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	**0.00**	0.00
	2828	16.67	16.67	16.67	16.67	**0.00**	13.33	**0.00**	15.00	16.67	16.67	16.67	16.67	16.67	16.67	16.67	16.67
Gunther35	41	5.56	11.11	11.11	12.78	5.56	13.89	5.56	13.89	5.56	11.67	5.56	11.67	11.11	12.22	**0.00**	9.44
	44	6.67	12.00	6.67	10.00	6.67	10.00	6.67	12.67	6.67	11.33	6.67	9.33	6.67	12.00	**0.00**	8.67
	49	8.33	9.17	8.33	8.33	**0.00**	7.50	8.33	9.17	8.33	10.00	8.33	8.33	8.33	8.33	**0.00**	8.33
	54	9.09	9.09	9.09	9.09	**0.00**	8.18	9.09	9.09	9.09	9.09	9.09	9.09	9.09	9.09	**0.00**	8.18
	61	**0.00**	6.00	**0.00**	6.00	**0.00**	3.00	**0.00**	2.00	**0.00**	4.00	**0.00**	6.00	**0.00**	1.00	**0.00**	3.00
	69	**0.00**	3.33	**0.00**	1.11	**0.00**	1.11	**0.00**	2.22	**0.00**	0.00	**0.00**	5.56	**0.00**	4.44	**0.00**	1.11
	81	**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	**0.00**	0.00
Kilbridge45	57	**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	**0.00**	0.00
	79	**0.00**	2.50	**0.00**	2.50	**0.00**	5.00	**0.00**	1.25	**0.00**	1.25	**0.00**	5.00	**0.00**	1.25	**0.00**	1.25
	92	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	1.43	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
	110	**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	**0.00**	0.00
	138	**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	**0.00**	0.00
	184	**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	**0.00**	0.00
Hahn53	2004	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	1.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00
	2338	**0.00**	11.25	**0.00**	7.50	**0.00**	10.00	**0.00**	8.75	**0.00**	7.50	**0.00**	3.75	**0.00**	8.75	**0.00**	5.00
	2806	**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	**0.00**	0.00
	3507	**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	**0.00**	0.00
	4676	**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	**0.00**	0.00
Warnecke58	54	**0.00**	1.90	**0.00**	1.67	**0.00**	1.43	**0.00**	1.90	**0.00**	1.43	**0.00**	1.19	**0.00**	1.43	**0.00**	1.67
	56	2.63	3.95	**0.00**	2.63	**0.00**	2.89	2.63	3.68	2.63	3.68	2.63	3.16	2.63	3.42	**0.00**	2.63
	65	3.23	3.87	3.23	4.19	**0.00**	3.23	**0.00**	2.26	**0.00**	2.58	**0.00**	1.94	3.23	3.87	**0.00**	3.23
	71	3.70	6.30	3.70	6.67	3.70	5.93	**0.00**	5.56	3.70	7.04	3.70	7.04	3.70	7.41	3.70	6.30
	82	4.55	8.18	9.09	9.09	4.55	8.64	4.55	8.64	9.09	9.09	9.09	9.09	9.09	9.09	4.55	8.64
	92	5.00	6.50	5.00	7.50	**0.00**	6.00	**0.00**	5.50	5.00	5.50	5.00	6.00	5.00	5.50	5.00	5.00
	111	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	**0.00**	5.62

Table 12
Comparison results of ACS variants for large-scaled instances.

Instances	*CT*	ACS1		ACS3		ACS5		ACS7		ACS9		ACS11		ACS13		ACS16	
		Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
Tonge70	176	**0.00**	3.08	**0.00**	3.46	**0.00**	2.69	3.85	4.23	**0.00**	3.46	**0.00**	3.46	**0.00**	3.46	**0.00**	3.08
	364	**0.00**	2.50	**0.00**	5.83	**0.00**	2.50	**0.00**	4.17	**0.00**	5.00	**0.00**	2.50	**0.00**	3.33	**0.00**	1.67
	410	**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	**0.00**	0.00
	468	**0.00**	6.67	**0.00**	5.56	**0.00**	10.00	**0.00**	7.78	**0.00**	4.44	**0.00**	7.78	**0.00**	7.78	**0.00**	3.33
	527	**0.00**	3.75	**0.00**	3.75	**0.00**	2.50	**0.00**	1.25	**0.00**	3.75	**0.00**	3.75	**0.00**	2.50	**0.00**	3.75
Weemag75	28	1.47	2.50	1.47	2.21	1.47	2.35	1.47	1.91	1.47	2.35	1.47	2.50	1.47	2.35	1.47	2.21
	32	1.61	2.90	1.61	2.58	1.61	2.26	**0.00**	2.10	1.61	2.58	1.61	2.42	1.61	2.42	1.61	2.74
	35	1.64	3.11	1.64	2.95	**0.00**	2.46	**0.00**	2.46	1.64	2.62	1.64	2.46	**0.00**	2.30	1.64	2.30
	38	1.67	3.50	1.67	3.17	1.67	3.00	1.67	3.00	**0.00**	2.50	1.67	2.83	1.67	2.67	1.67	2.83
	40	1.67	3.00	1.67	3.17	1.67	2.83	1.67	2.67	**0.00**	2.67	1.67	3.17	1.67	2.17	**0.00**	2.67
	47	2.70	3.78	**0.00**	2.43	2.70	2.97	2.70	3.24	**0.00**	3.51	**0.00**	3.51	2.70	3.24	**0.00**	2.70
	56	3.23	3.23	3.23	3.23	3.23	3.55	3.23	3.23	3.23	3.55	3.23	3.23	3.23	3.23	**0.00**	2.90
Arcus83	5048	**0.00**	1.67	**0.00**	0.56	**0.00**	0.00	**0.00**	0.56	**0.00**	0.00	**0.00**	0.00	**0.00**	1.11	**0.00**	0.00
	5853	6.67	6.67	**0.00**	4.67	**0.00**	4.67	**0.00**	3.33	**0.00**	4.00	**0.00**	5.33	**0.00**	5.33	**0.00**	4.67
	6842	**0.00**	0.77	**0.00**	0.00	**0.00**	0.77	**0.00**	1.54	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.77
	7571	**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	**0.00**	0.00
	8412	10.00	10.00	10.00	10.00	10.00	10.00	**0.00**	9.00	**0.00**	9.00	10.00	10.00	**0.00**	9.00	10.00	10.00
	8998	**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	**0.00**	0.00
	10,816	**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	**0.00**	0.00
Lutz89	75	**0.00**	2.14	**0.00**	2.14	**0.00**	2.14	**0.00**	1.79	**0.00**	2.50	**0.00**	1.79	**0.00**	1.43	**0.00**	2.14
	83	**0.00**	3.60	**0.00**	3.20	**0.00**	3.60	**0.00**	3.60	**0.00**	3.60	**0.00**	3.20	**0.00**	3.20	4.00	4.00
	92	**0.00**	3.18	**0.00**	2.73	**0.00**	5.00	**0.00**	4.09	**0.00**	3.18	**0.00**	4.09	4.55	4.55	**0.00**	2.27
	110	5.56	5.56	**0.00**	3.89	5.56	6.11	5.56	6.67	5.56	6.11	5.56	5.56	**0.00**	5.56	**0.00**	4.44
	118	**0.00**	0.59	**0.00**	0.00	**0.00**	0.59	**0.00**	1.18	**0.00**	1.18	**0.00**	0.59	**0.00**	2.35	**0.00**	0.00
	137	**0.00**	6.43	**0.00**	6.43	**0.00**	3.57	**0.00**	4.29	**0.00**	4.29	**0.00**	5.71	**0.00**	5.00	**0.00**	6.43
	150	**0.00**	5.38	**0.00**	3.85	7.69	7.69	**0.00**	6.92	7.69	7.69	**0.00**	6.92	**0.00**	6.15	**0.00**	6.15
Mukherje94	176	**0.00**	5.71	3.57	6.43	3.57	6.07	3.57	6.43	3.57	6.07	3.57	6.07	3.57	6.07	3.57	4.64
	183	3.70	5.56	3.70	5.56	**0.00**	3.33	3.70	5.19	3.70	4.44	3.70	4.44	**0.00**	4.44	3.70	4.81
	192	3.85	3.85	**0.00**	3.46	**0.00**	3.08	**0.00**	3.46	**0.00**	3.46	3.85	3.85	**0.00**	3.46	3.85	3.85
	201	4.00	4.00	**0.00**	4.00	4.00	4.00	**0.00**	3.20	4.00	4.00	**0.00**	3.60	**0.00**	3.60	4.00	4.00
	211	4.17	4.17	**0.00**	3.75	**0.00**	3.75	4.17	4.17	**0.00**	2.92	**0.00**	3.75	4.17	4.17	**0.00**	2.92
	324	**0.00**	2.67	**0.00**	2.67	**0.00**	3.33	**0.00**	3.33	**0.00**	1.33	**0.00**	3.33	**0.00**	0.67	**0.00**	0.67
	351	**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	**0.00**	0.00
Arcus111	5755	3.13	5.31	3.13	4.38	3.13	5.63	**0.00**	4.69	3.13	5.63	3.13	5.94	3.13	5.63	3.13	4.69
	8847	**0.00**	2.50	**0.00**	1.00	**0.00**	2.00	**0.00**	1.50	**0.00**	1.00	**0.00**	2.50	**0.00**	2.50	**0.00**	2.00
	10,027	**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	**0.00**	0.00
	10,743	**0.00**	0.59	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.59	**0.00**	1.18	**0.00**	0.00	**0.00**	0.00
	11,378	**0.00**	5.33	**0.00**	4.67	**0.00**	4.00	**0.00**	2.67	**0.00**	4.67	**0.00**	3.33	**0.00**	1.33	**0.00**	1.33
	17,067	**0.00**	7.00	**0.00**	8.00	**0.00**	7.00	**0.00**	5.00	**0.00**	5.00	**0.00**	6.00	**0.00**	7.00	**0.00**	6.00
Barthol148	84	1.59	2.70	1.59	2.54	1.59	2.54	1.59	2.22	1.59	2.54	1.59	1.90	**0.00**	2.22	**0.00**	1.59
	99	1.96	1.96	**0.00**	1.76	**0.00**	2.55	**0.00**	2.16	**0.00**	2.75	1.96	2.35	1.96	2.55	**0.00**	1.37
	101	**0.00**	1.40	2.00	2.20	**0.00**	1.80	**0.00**	1.60	**0.00**	1.80	**0.00**	1.80	**0.00**	1.40	**0.00**	0.80
	115	2.33	2.79	2.33	4.19	2.33	4.19	2.33	3.49	2.33	3.72	2.33	4.19	2.33	3.26	2.33	2.79
	118	2.38	2.38	2.38	3.81	2.38	3.57	2.38	3.10	2.38	2.86	2.38	3.10	2.38	3.10	**0.00**	2.14
	157	3.23	3.23	3.23	3.55	3.23	3.23	3.23	3.23	**0.00**	2.90	3.23	3.23	3.23	3.23	3.23	3.23
	170	**0.00**	1.72	**0.00**	2.76	**0.00**	2.07	**0.00**	1.72	**0.00**	2.76	**0.00**	2.76	**0.00**	2.07	**0.00**	1.03
Scholl297	1394	**0.00**	1.50	**0.00**	1.50	1.67	2.33	1.67	2.67	1.67	2.83	1.67	1.67	1.67	1.67	**0.00**	1.00
	1483	**0.00**	1.61	1.79	2.14	1.79	2.68	1.79	2.14	**0.00**	2.86	1.79	2.14	1.79	2.68	**0.00**	1.61
	1659	2.04	2.86	2.04	3.67	2.04	3.47	2.04	3.47	2.04	3.88	2.04	2.45	2.04	3.27	2.04	2.24
	1834	2.27	4.32	2.27	4.32	2.27	4.09	2.27	3.64	2.27	3.64	2.27	3.64	2.27	4.32	**0.00**	2.27
	2049	**0.00**	1.50	**0.00**	1.50	**0.00**	1.50	**0.00**	2.00	2.50	2.50	**0.00**	0.50	**0.00**	1.50	**0.00**	0.00
	2322	**0.00**	1.43	**0.00**	1.71	2.86	2.86	2.86	2.86	2.86	3.14	**0.00**	2.29	**0.00**	2.57	**0.00**	0.00
	2488	3.13	4.38	3.13	4.06	6.25	6.25	3.13	5.31	3.13	5.63	3.13	3.44	3.13	4.06	**0.00**	2.81
	2787	**0.00**	3.10	**0.00**	3.10	3.45	3.45	3.45	3.45	3.45	3.45	**0.00**	2.76	**0.00**	3.10	**0.00**	2.41

As illustrated in Table 10, 50 instances in ACS1, 51 instances in ACS3, 51 instances in ACS5, 51 instances in ACS7, 50 instances in ACS9, 52 instances in ACS11, 50 instances in ACS13 and 50 instances in ACS16 get the minimum solution in total 52 of small-scaled instances. Although other instances do not get the minimum value, they are near to the minimum value. This means that ACS algorithm embedded with these heuristic rules are an effective way to solve the small-scaled instances of TSALBP with multi-manned workstations.

As for the middle-scaled instances, it is observed from Table 11 that 23 instances in ACS1, 21 instances in ACS3, 29 instances in ACS5, 28 instances in ACS7, 24 instances in ACS9, 22 instances in ACS11, 24 instances in ACS13 and 31 instances in ACS16 are superior to others in total 38 of middle-scaled instances. And the average values of the minimum *RPD* of ACS variants are equal to 3.23, 4.08, 1.80, 2.02, 3.05, 3.47, 3.11 and 1.75 respectively. If using the number of times to get the minimum value to rank the ACS variants, ACS16 performs best whereas ACS3 has the worst performance. If using the average values of the minimum *RPD* to rank variants, ACS16 still performs best, and ACS5 follows whereas ACS3 is worst. Hence, it is concluded that ACS16 is significantly more advanced in the middle-scaled instances of TSALBP with multi-manned workstations.

Regarding to the results of large-scaled instances shown in Table 12, 31 instances in ACS1, 35 instances in ACS3, 31 instances in ACS5, 33 instances in ACS7, 34 instances in ACS9, 32 instances in ACS11, 35 instances in ACS13 and 40 instances in ACS16 reach to the minimum value in total 54 of large-scaled instances. And in this case the mean of the minimum *RPD* of ACS variants are equal to 1.37, 0.97, 1.41, 1.08, 1.11, 1.18, 0.90 and 0.86 respectively. It is clearly that ACS16 outperforms other ACS variants in both the number of times to get the minimum value and the mean of the minimum *RPD*.

All in all, as the instance scale increases, ACS16 is not only more outstanding but also stable. The excellent performance of ACS16 is due to the following reasons. The heuristic rules 1–15 just provide one type information when helping the algorithm select a task. In this way of selection, if the ants in front find worse paths, the ants in the tail could acquire the wrong information and then generate a locally optimal solution. However, the heuristic rule 16 randomly select one rule of the first 15 heuristic rules to help each ant select a task in each iteration. This stochastic mechanism not only reserve all heuristic information to guide ants to select tasks, but also reduce the risk of miscommunication.

Moreover, statistical analysis is carried out to further confirm the statistically significant difference between the ACS variants. A multiple ANOVA test is carried out where *RPD* and algorithm type are regarded as the response variable and controlled factor respectively. Figs. 5 and 6 depict the means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all ACS variants under $\rho = 10$ and 20 stopping criteria respectively. In these figures, 144 comparison instances are involved, and the ACS variants is ranked according to the minimum *RPD*. And ANOVA results for the heuristic rules are shown in Table 13. As observed in Table 13, the *F*-ratio is 0.19 and *P*-value is 1.00, which demonstrates that the type of heuristic rules has little effect on the performance of ACS algorithm. However, it still is observed from the Figs. 5 and 6 that ACS16 performs best and ACS4 has worst performance.

The above comparison experiments compare ACS variants with each other and lead to a conclusion that ACS16 is superior to other variants. The deviation ratio is employed again to further explore the quality of ACS16. Fig. 7 shows the deviation ratio of ACS16 and lower bound under 144 comparison instances. As seen, more than ten instances reach the low bound and almost all instances are in the 0–0.5 range. In Section 4.2, we conclude that the algorithm can basically be considered to own better performance if the deviation ratio is kept within 0.5. Hence, ACS16 have great performance in minimize the total number of operators and workstations of TSALBP with multi-manned workstations.

5.5. Analysis of the effectiveness of the local search

In this section, a comparison experiment between ACS_LS and ACS is designed to test the effectiveness of the proposed referenced local search. Similar to the above experiment, ACS and ACS_LS algorithms are solved under the same stopping criteria, a CPU time limit of $n \times n \times \rho$ milliseconds (in this section two values of ρ are considered: 10 and 20) for a fair comparison. And Each algorithm is run on each one of the 144 comparison instances with two stopping criteria 10 times. A total of 5760 experiments are collected and analyzed. For the lack of space, this section does not present the specific data of every instance and just depicts the means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for ACS_LS and ACS under $\rho = 10$ and 20 stopping criteria. The compared results are depicted in Fig. 8. Through the Figure, it can be observed that the ACS_LS outperforms ACS under two stopping criteria. These results demonstrate the effectiveness of the proposed referenced local search.

6. Conclusions and future work

Multi-manned workstations are widely used to assemble large size and high volume of products to reduce the length of assembly line and the amount of throughput time. The consideration of dual attributes (time and space) of tasks in assembly line balancing problem is in line with the production of real enterprises. However, there is no research regarding multi-manned workstations by including the time and space information for the assembly lines. This work first defines six kinds of constraints including task assignment constraint, precedence constraint, cycle time constraint, sequencing constraint, space constraint and variable constraint to formulate a new problem: the time and space assembly line balancing problem with multi-manned workstations. Besides, due to the NP-hardness nature of this problem, a memetic ACO algorithm is developed to tackle the middle- and large-scaled instances while a MILP approach is used for small instances.

Three ACO improvement strategies, including a new solution generation method embedded with 16 heuristic rules, a new pheromone release strategy, and the best solution updating method, are designed to enhance the performance of proposed the memetic ACS algorithm. We carried out a comprehensive statistical and computational results to obtain three main conclusions:

(1) The new problem is necessary and realistic in modern assembly enterprises.
(2) For small-scale TSALBP with multi-manned workstations, the proposed MILP model can be solved by GAMS/CPLEX to obtain the optimal task assignment plan.
(3) For middle- and large-scale instances, the proposed memetic ACS_LS algorithm embedded with the 16-th heuristic rule obtains the highest performance. Therefore, we suggest its use to obtain the near-optimal or optimal task assignment plan.

Future research could extend the current model to consider uncertain

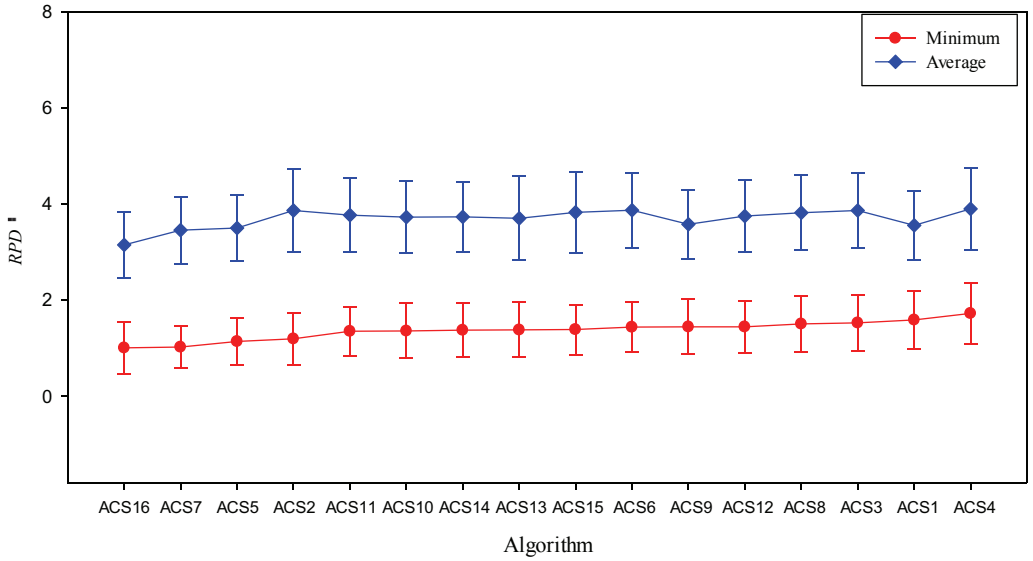

Fig. 5. Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all ACS variants and $\rho = 10$ stopping criterion.

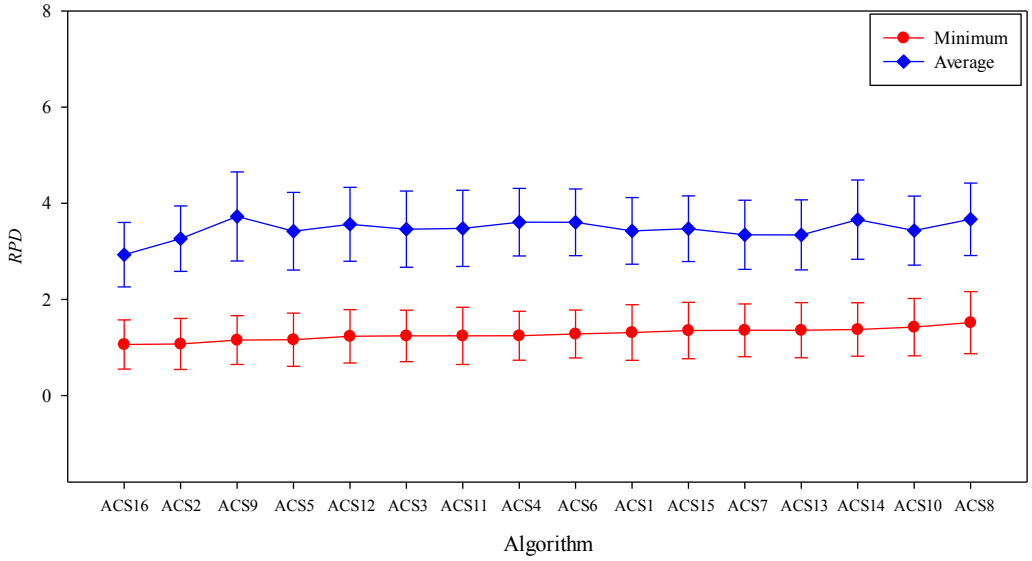

Fig. 6. Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all ACS variants and $\rho = 20$ stopping criterion.

Table 13
ANOVA results for the heuristic rules.

Sources	df	Type III sum of squares	Mean square	*F*-ratio	*P*-value
Rules	15	33.2	2.212	0.19	1.000
Error	2288	26118.1	11.415		
Total	2303	26151.2			

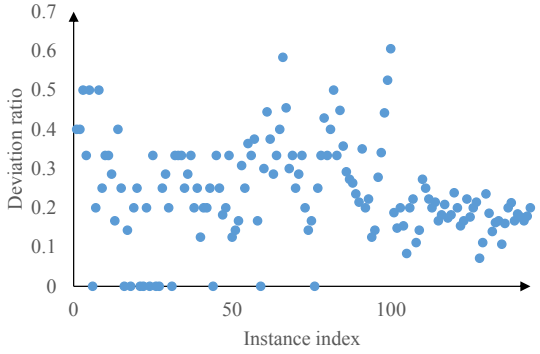

Fig. 7. Deviation ratio of the ACS_16 and lower bound.

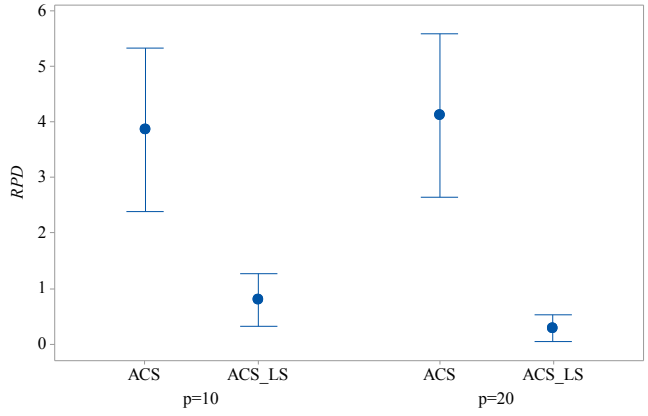

Fig. 8. Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for ACS_LS and ACS.

demand with multi-manned workstations. The multi-manned problem can also be extended by simultaneously considering different objectives (e.g., cycle time and global area) in a multi-objective optimization approach (Chica et al., 2010). Besides, future research could be also focused on designing a hybrid approach where we combine a heuristic to obtain a good initial solution and then we apply both the metaheuristic and CPLEX methods.

CRediT authorship contribution statement

Zikai Zhang: Methodology, Writing - original draft. **Qiuhua Tang:** Writing - review & editing. **Manuel Chica:** Writing - review & editing.

Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 51875421). M. Chica is also supported through the Ramón y Cajal program (RYC-2016-19800) and by Spanish Ministry of Science, Innovation and Universities under the EXASOCO project (PGC2018-101216-B-I00).

References

Babazadeh, H., & Javadian, N. (2018). A novel meta-heuristic approach to solve fuzzy multi-objective straight and U-shaped assembly line balancing problems. *Soft Computing*.
Bautista, J., & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research, 177*, 2016–2032.
Bautista, J., & Pereira, J. (2011). Procedures for the time and space constrained assembly line balancing problem. *European Journal of Operational Research, 212*, 473–481.
Baykasoglu, A., & Dereli, T. (2008). Two-sided assembly line balancing using an ant-colony-based heuristic. *The International Journal of Advanced Manufacturing Technology, 36*, 582–588.
Boysenaab, N. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research, 183*, 674–693.
Buyukozkan, K., Kucukkoc, I., Satoglu, S. I., & Zhang, D. Z. (2016). Lexicographic bottleneck mixed-model assembly line balancing problem: Artificial bee colony and tabu search approaches with optimised parameters. *Expert Systems with Applications, 50*, 151–166.
Chen, Y.-Y. (2017). A hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines. *Journal of Manufacturing Systems, 42*, 196–209.
Chen, Y.-Y., Cheng, C.-Y., & Li, J.-Y. (2018). Resource-constrained assembly line balancing problems with multi-manned workstations. *Journal of Manufacturing Systems, 48*, 107–119.
Chica, M., Bautista, J., Cordón, Ó., & Damas, S. (2016). A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand. *Omega, 58*, 55–68.

Chica, M., Cordón, Ó., & Damas, S. (2011). An advanced multiobjective genetic algorithm design for the time and space assembly line balancing problem. *Computers & Industrial Engineering, 61*, 103–117.
Chica, M., Cordón, Ó., Damas, S., & Bautista, J. (2010). Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. *Information Sciences, 180*, 3465–3487.
Chica, M., Cordón, Ó., Damas, S., & Bautista, J. (2012). Multiobjective memetic algorithms for time and space assembly line balancing. *Engineering Applications of Artificial Intelligence, 25*, 254–273.
Cordón García, O., Herrera Triguero, F., & Stützle, T. (2002). A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing, 9*.
Delice, Y., Kızılkaya Aydoğan, E., Özcan, U., & İlkay, M. S. (2017). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing, 28*, 23–36.
Dimitriadis, S. G. (2006). Assembly line balancing and group working: A heuristic procedure for workers' groups operating on the same product and workstation. *Computers & Operations Research, 33*, 2757–2774.
Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*, 53–66.
Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B, Cybernetics: A publication of the IEEE Systems, Man, and Cybernetics Society, 26*, 29–41.
Elmi, A., & Topaloglu, S. (2017). Cyclic job shop robotic cell scheduling problem: Ant colony optimization. *Computers & Industrial Engineering*.
Fattahi, P., Roshani, A., & Roshani, A. (2010). A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology, 53*, 363–378.
Karp, R. M. (2010). Reducibility among combinatorial problems. In M. Jünger, T. M. Liebling, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, … L. A. Wolsey (Eds.), *50 years of integer programming 1958–2008: from the early years to the state-of-the-art* (pp. 219–241). Berlin Heidelberg, Berlin, Heidelberg: Springer.
Kellegöz, T. (2016). Assembly line balancing problems with multi-manned stations: A new mathematical formulation and Gantt based heuristic method. *Annals of Operations Research*.
Kellegöz, T., & Toklu, B. (2012). An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations. *Computers & Operations Research, 39*, 3344–3360.
Kellegöz, T., & Toklu, B. (2015). A priority rule-based constructive heuristic and an improvement method for balancing assembly lines with parallel multi-manned workstations. *International Journal of Production Research, 53*, 736–756.
Kim, Y. K., Kim, J. Y., & Kim, Y. (2000). A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. *Applied Intelligence, 13*, 247–258.
Kucukkoc, I., & Zhang, D. Z. (2016). Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach. *Computers & Industrial Engineering, 97*, 58–72.
Kurdi, M. (2018). Ant colony system with a novel Non-DaemonActions procedure for multiprocessor task scheduling in multistage hybrid flow shop. *Swarm and Evolutionary Computation*.
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. *Journal of Cleaner Production, 135*, 508–522.
Li, M., Tang, Q., Zheng, Q., Xia, X., & Floudas, C. A. (2017). Rules–based heuristic approach for the U-shaped assembly line balancing problem. *Applied Mathematical Modelling, 48*, 423–439.
Li, Y., Wang, H., & Yang, Z. (2018). Type II assembly line balancing problem with multi-operators. *Neural Computing and Applications*.
Lopes, T. C., Pastre, G. V., Michels, A. S., & Magatão, L. (2019). Flexible multi-manned assembly line balancing problem: Model, heuristic procedure, and lower bounds for line length minimization. *Omega*.
Michels, A. S., Lopes, T. C., Sikora, C. G. S., & Magatão, L. (2019). A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem. *European Journal of Operational Research*.
Miltenburg, J. (2001). U-shaped production lines: A review of theory and practice. *International Journal of Production Economics, 70*, 201–214.
Mosadegh, H., Ghomi, S. M. T. F., & Süer, G. A. (2019). Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and Q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research*.
Mukund Nilakantan, J., Huang, G. Q., & Ponnambalam, S. G. (2015). An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *Journal of Cleaner Production, 90*, 311–325.
Nourmohammadi, A., Fathi, M., Zandieh, M., & Ghobakhloo, M. (2019). A water-flow like algorithm for solving U-shaped assembly line balancing problems. *IEEE Access*, 1 1.
Oksuz, M. K., Buyukozkan, K., & Satoglu, S. I. (2017). U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics. *Computers & Industrial Engineering, 112*, 246–263.
Pan, Q.-K., Gao, L., Wang, L., Liang, J., & Li, X.-Y. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications*.
Rada-Vilela, J., Chica, M., Cordón, Ó., & Damas, S. (2013). A comparative study of multi-objective ant colony optimization algorithms for the time and space assembly line balancing problem. *Applied Soft Computing, 13*, 4370–4382.
Roshani, A., & Ghazi Nezami, F. (2017). Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach. *Assembly Automation, 37*.
Roshani, A., & Giglio, D. (2016). Simulated annealing algorithms for the multi-manned assembly line balancing problem: Minimising cycle time. *International Journal of Production Research*.
Roshani, A., Roshani, A., Roshani, A., Salehi, M., & Esfandyari, A. (2013). A simulated annealing algorithm for multi-manned assembly line balancing problem. *Journal of Manufacturing Systems, 32*, 238–247.
Şahin, M., & Kellegöz, T. (2016). Increasing production rate in U-type assembly lines with sequence-dependent set-up times. *Engineering Optimization*, 1–19.
Şahin, M., & Kellegöz, T. (2019). A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations. *Computers & Industrial Engineering*.
Samouei, P., & Ashayeri, J. (2019). Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations. *Applied Mathematical Modelling*.
Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*, 666–693.
Simaria, A. S., & Vilarinho, P. M. (2009). 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering, 56*, 489–506.
Yilmaz, H., & Yilmaz, M. (2015). Multi-manned assembly line balancing problem with balanced load density. *Assembly Automation, 35*, 137–142.
Zhang, Z., Tang, Q., Han, D., & Li, Z. (2018). Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment. *Neural Computing and Applications, 31*, 7501–7515.
Zhang, Z., Tang, Q., Li, Z., & Zhang, L. (2018). Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems. *International Journal of Production Research, 57*, 5520–5537.
Zhang, Z., Tang, Q., & Zhang, L. (2019). Mathematical model and grey wolf optimization for low-carbon and low-noise U-shaped robotic assembly line balancing problem. *Journal of Cleaner Production, 215*, 744–756.
Zhao, X., Hsu, C.-Y., Chang, P.-C., & Li, L. (2016). A genetic algorithm for the multi-objective optimization of mixed-model assembly line based on the mental workload. *Engineering Applications of Artificial Intelligence, 47*, 140–146.