

# A modified ant colony optimization algorithm for multi-objective assembly line balancing

Yu-guang Zhong<sup>1</sup> · Bo Ai<sup>1</sup>

Published online: 1 July 2016  
© Springer-Verlag Berlin Heidelberg 2016

**Abstract** In this paper, a novel ant colony optimization algorithm called modified ant colony optimization algorithm (MACO) is proposed for multi-objective single-model assembly line balancing problem (SALBP). The proposed MACO presents a novel heuristic information combined with subsequent task number and deviation time that can guide ants to find better solutions for SALBP. The proposed MACO also adopts three assignment methods (i.e., forward, backward and local rebalancing assignment methods) and stratified sequential algorithm combined with Pareto-optimal front as multi-objective decision. The objectives of SALBP are to minimize the number of workstations, maximize assembly line efficiency and minimize workload variation among workstations. In the latter part of the paper, the proposed MACO has been applied to solve Scholl benchmark problems which include both small-size and large-size problems. The performance of the proposed MACO has been compared with the multi-objective genetic algorithm and the multiple assignment genetic algorithm and has obtained improved results in many test problems.

**Keywords** Ant colony optimization algorithm · Assembly line balancing · Multi-objective · Heuristic information

## 1 Introduction

An assembly line is a continuous production line which consists of materials and workstations combined by conveyor belts. It is a kind of system which can contact men with machines closely and efficiently. Assembly is not only the final step, but also the most important process of the manufacturing system. Planning the layout of workstations is the first task, while building an assembly line and assembly line balancing problem (ALBP) is especially important. Balancing the assembly line can keep the production system continuous and fluent. It also has a great influence on the productivity of the manufacturing system.

The assembly line balancing problem is a kind of optimization problems which distribute tasks to workstations satisfying the precedence constraint and optimizing some objectives. It can be divided into simple model assembly line balancing problem (SMALBP) and mixed-model assembly line problem (MMALBP) according to the types of product. Simple assembly line balancing problem (SALBP) is the best known ALBP. It can be divided into three types: SALBP-1, SALBP-2 and SALBP-E (Becker and Scholl 2006). The SALBP-1 aims to minimize the number of the workstations within the cycle time. The SALBP-2 aims to minimize the cycle time, while the number of the workstations is known. SALBP-E aims to maximizing the line efficiency, while the number of workstations and the cycle time has been given.

ALBP has been proved as NP-hard problem of combination optimization problems (Karp 1972). If there are  $n$  tasks and  $m$  precedence constraints in an ALBP, and then there will be  $n!/2^m$  task sequences which are solutions of ALBP (Baybars 1986). The complexity of ALBP will be much higher as the size of problem increasing. Much more time have to be consumed for computing because the search space is so big.

---

Communicated by V. Loia.

✉ Yu-guang Zhong  
zhongyuguang@126.com

<sup>1</sup> College of Mechanical and Electrical Engineering, Harbin Engineering University, Harbin 150001, China

SALBP is generally solved by exact algorithm when it arises. Although we can obtain the optimal solution, we need too much time to calculate it as the problem size increased. So we usually use exact algorithm to solve small-size problems. In recent years, several meta-heuristic algorithms such as genetic algorithm, particle swarm optimization algorithm and ant colony optimization algorithm have been used to solve ALBP because of these algorithms' good performance on the optimization.

Yu and Yin (2010) put forward a new adaptive genetic algorithm to solve SALBP-1 in which crossover probability and mutation probability can be dynamic adjusted according to the fitness of individual. If the fitness of individual is high, the probability of crossover and mutation will be small. If not, the probability will be big. Gutiérrez and García-Magariño (2011) proposed a hybrid genetic algorithm which combines genetic algorithm with repair heuristics for flexible job-shop scheduling problems. Hou et al. (2014) proposed an improved genetic algorithm for balancing product family assembly line which is a mixed assembly line for a family of similar products. Petropoulos and Nearchou (2011) used an improved PSO to solve large-scale multi-objective SALBP. They developed a new method to generate and retain Pareto solution of multi-objective SALBP.

As the increasing of problems' scale, the complexity and the solution space of the problems increase exponentially. The shortage of based meta-heuristic algorithms for large-scale problems with big data has been revealed. Novel and efficient algorithms are in urgent need. Sfrent and Pop (2015) proposed an asymptotically optimal algorithm combined with first-come, first-served (FCFS) for asymptotic scheduling problems and achieve better performances. Vasile et al. (2015) proposed a resource-aware hybrid scheduling algorithm for tasks scheduling in heterogeneous distributed computing considering hierarchical clustering of the available resources into groups.

There are few papers with respect to ACO applied to solve ALBP as ACO is developed later than other meta-heuristic algorithms. But ACO has good performance on solving combinatorial optimization problems. ACO comparing with other swarm intelligence algorithms is just like transgenic technique comparing with hybridization technique that ACO can recognize which combination fragment is good guided by heuristic information and fed back by pheromone. However, other swarm intelligence algorithms can only recognize which individual is good. Although ACO may be slow in running because of much more information needed to be recognized and recorded than other swarm intelligence algorithms, fast emerging computer hardware can easily solve this problem.

In this paper, we develop a modified ant colony optimization algorithm (MACO) for multi-objective SALBP in which objectives are to minimize the number of work-

stations, maximize line efficiency and minimize workload variation. The proposed algorithm MACO develops a new heuristic information combining with subsequent task number and deviation time as expectation value of node which can guide ants search for the optimal solution effectively. The proposed algorithm also uses three assignment approaches: forward, backward and local rebalancing assignments to assign tasks to workstations simultaneously and calculate the value of objective function, respectively, and then choose the best assignment approach for ant. Using three assignment approaches can increase a task sequence potential effectively. After comparison among the ants using stratified sequential algorithm, we retain Pareto-optimal front ant.

The proposed MACO is very suitable to solve SALBP that it can search for better solutions rapidly and efficiently. The proposed MACO has been compared with multi-objective genetic algorithm (MOGA) (Hwang et al. 2008) and multiple assignment genetic algorithm (MA-GA) (Al-Hawari et al. 2015). All the three algorithms can get optimal solution on the first objective. The proposed MACO performs much better than the other two algorithms on the second and third objectives that it can get to a maximum of 26.58% improvement and 100% improvement on the second and third objective, respectively.

The structure of the rest paper is as follow: Sect. 2 describes the related work of ALBP. In Sect. 3, the formulation of multi-objective SALBP is given. We can see the detail of the proposed MACO in Sect. 4 and computational results in Sect. 5. Finally, there are discussions and conclusions in Sects. 6 and 7.

## 2 Related work

In the early years, when ALBP has just been presented, ALBP usually be solved by some exact algorithm, such as integer programming (Bowman 1960), linear programming (Salveson 1955), branch-and-bound approaches (Jackson 1956) and dynamic programming (Held et al. 1963). These exact algorithms can get optimal solution for small-size ALBP. However, these algorithms have to consume too much time which is unrealistic for calculating large-size problems.

In recent years, much more scholars and researchers pay attention to meta-heuristic algorithms for solving ALBP. The solutions obtained by meta-heuristic algorithms for solving ALBP may not be the optimal solution. But meta-heuristic algorithms can get acceptable solutions in reasonable time no matter small-size or large-size problems.

Lv (2011) used an improved particle swarm optimization algorithm (PSO) to solve SALBP-1. In this paper, a new encoding method has been proposed to ensure that a task sequence is feasible. Each particle which represents a feasible task sequence can search for the best solution based on

the optimization mechanism of the PSO. [Dou et al. \(2013\)](#) proposed a novel feasible task sequence oriented PSO that directly records a feasible task sequence by a particle for solving SALBP-1. However, the algorithms presented by the literature above can only solve single objective of SALBP-1.

[Kilincci \(2011\)](#) proposed a new heuristic algorithm based on Petri net approach. The proposed algorithm named firing sequences backward algorithm has a good performance to solve SALBP-1. [Hamta et al. \(2013\)](#) proposed a hybrid PSO algorithm that combines PSO algorithm with variable neighborhood search to solve multi-objective assembly balancing problems with flexible operation times, sequence-dependent setup times and learning effect. [Li et al. \(2014\)](#) proposed a multi-objective teaching–learning-based optimization algorithm for balancing two-sided assembly line.

[Bautista and Pereira \(2007\)](#) used an improved ant colony optimization algorithm (ACO) to solve a time and space constrained assembly line balancing problem (TSALBP). In this paper, they have established mathematical model for TSALBP in the first place and put forward a new ACO with fusion ideas and got a good result in the end. [Yagmahan \(2011\)](#) proposed a multi-objective ant colony optimization algorithm for mixed-model assembly line balancing problems and obtains better results.

[Hwang et al. \(2008\)](#) presented a multi-objective genetic algorithm (MOGA) for straight line and *U*-shaped assembly line balancing problems. Then, the proposed MOGA was applied for Scholl benchmark problems within different cycle time. It could get the optimal solution for the first objective of minimum number of workstations. However, the solutions for the second and the third objective were unsatisfactory. [Al-Hawari et al. \(2015\)](#) used a multiple assignment genetic algorithm (MA-GA) to solve multi-objective SALBP. This algorithm assigns tasks to the workstations in three ways: forward, backward and bidirectional assignment. This improved genetic algorithm has better performance than MOGA presented in the previous literature.

In this paper, we propose a modified ant colony algorithm with a novel heuristic information and three kinds of assignment approach to solve multi-objective SALBP. The proposed MACO has been compared with MOGA and MA-GA for several Scholl benchmark problems. The results is much better than the other algorithms.

### 3 Assembly line balancing problem

#### 3.1 Basic description of SALBP

Figure 1 shows a precedence graph of Bowman problem model ([Scholl 1993](#)) which presents the precedence constraints of the tasks. There are a total of 8 tasks which are as nodes in Fig. 1. The arrows mean the precedence constraints

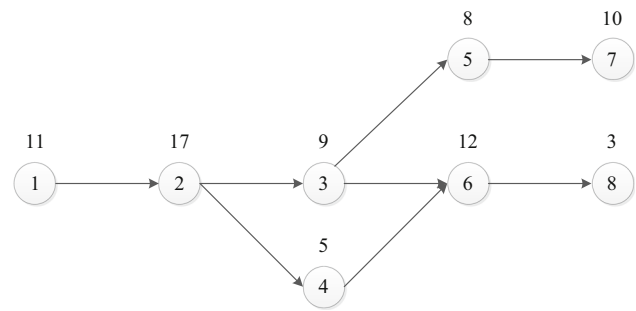


Fig. 1 A precedence graph

among the nodes. Only if all the immediate predecessor tasks of a task have been done, this task can be done.

Figure 2 shows an assumption of tasks assignment. Each task must be assigned to workstations front to back within precedence constraint and cycle time constraint.

In this paper, we research simple assembly line balancing problem. This SALBP has determinate task time, precedence constraints and a given cycle time. The objectives are to minimize the number of workstations, maximize line efficiency and minimize workload variation, simultaneously.

The characteristics of SALBP are as follows ([Baybars 1986](#); [Boysen et al. 2007](#)):

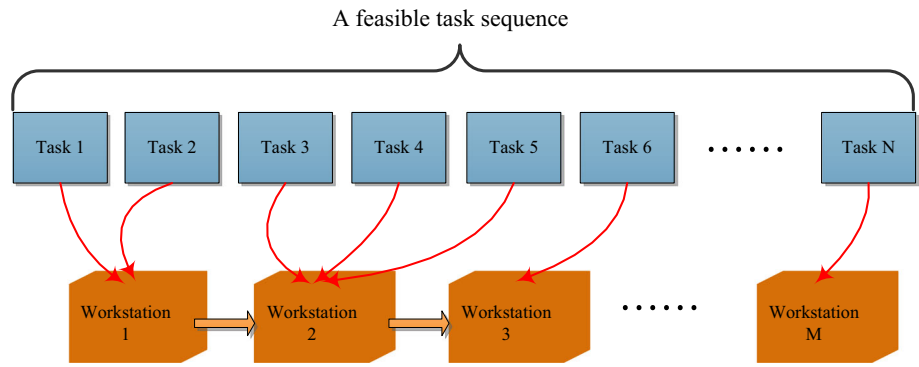
- There is only one type of product on assembly line.
- Each task can be assembled in any one of workstations.
- Each task must not be assembled in more than one workstation.
- Task time is determinate.
- All the tasks must be assigned to workstations.
- Only if all the immediate predecessor tasks of a task have been done, this task can be done.
- The sum of the task time in a same workstation must not be greater than cycle time.

#### 3.2 Mathematical formulation

The notation that exists in mathematical model can be summarized as follows:

$i, j$	Task number, $i, j = 1, 2, 3, \dots, N$
$k$	Workstation number, $k = 1, 2, 3, \dots, M$
$N$	The number of tasks
$M$	The number of workstations
$C$	Theoretical cycle time which is given as precondition
$t_i$	The process time of task $i$
$S_k$	The set of tasks which are assigned to the $k$ th workstation
$T(S_k)$	The sum time of tasks which are assigned to the $k$ th workstation

**Fig. 2** Task assignment illustration



- $C_a$  Actual cycle time (i.e.,  $C_a = \max_{1 \leq k \leq M} T(S_k)$ )
- $Prec(i)$  The set of tasks which are the immediate predecessor tasks of task  $i$
- $U_k$  Utilization of the  $k$ th workstation (i.e.,  $U_k = T(S_k) / C_a$ )
- $\bar{U}$  The average of all the workstation utilization, (i.e.,  $\bar{U} = \sum_{k=1}^M U_k / M$ )
- $E$  Assembly line efficiency (i.e.,  $E = \sum_{i=1}^N t_i / (m \cdot C_a)$ )
- $W_v$  Workload variation among workstations (i.e.,  $W_v = \sqrt{\sum_{k=1}^M (U_k - \bar{U})^2 / M}$ )
- $x_{ik}$  If task  $i$  is assigned to the  $k$ th workstation,  $x_{ik} = 1$ , otherwise  $x_{ik} = 0$

3.2.1 Mathematical model

The mathematical model for SALBP is composed of three objective functions as follow (Al-Hawari et al. 2015):

$$\min M = \sum_{k=1}^N \max_{1 \leq i \leq N} (x_{ik}) \tag{1}$$

$$\max E = \sum_{i=1}^N t_i / (m \cdot C_a) \tag{2}$$

$$\min W_v = \sqrt{\sum_{k=1}^M (U_k - \bar{U})^2 / M} \tag{3}$$

Subject to:

$$x_{ik} = 0 \text{ or } 1, \quad \forall (i, k) \tag{4}$$

$$\sum_{k=1}^M x_{ik} = 1, \quad \forall i \tag{5}$$

$$\sum_{k=1}^M kx_{ik} \leq \sum_{k=1}^M kx_{jk}, \quad \forall (i, j), \quad i \in Prec(j) \tag{6}$$

$$T(S_k) \leq C \tag{7}$$

The first objective function (1) is to minimize the number of workstations. The second objective function (2) is to maximize assembly line efficiency, and the third objective function (3) is to minimize the workload variation among the workstations.

Formulas (4–7) are the constraints of SALBP. Constraint (5) combined with constraint (4) means that each task can be assigned into just only one workstation. Constraint (6) means that task  $i$  prior to task  $j$  cannot be assigned to the workstation whose number is bigger than that of the workstation which task  $j$  is assigned. The last constraint (7) means that the work time of any workstations cannot be bigger than the theoretical cycle time  $C$ .

4 Proposed modified ant colony optimization algorithm

Ant colony optimization algorithm is created by imitating ants foraging. There must be a lot of paths between ant hole and food, and most of the ants will eventually focus on the shortest path.

We usually consider a problem as a directed graph when we use ant colony optimization algorithm to solve a problem. The ant travels in the directed graph within precedence constraints and a feasible solution will be generated after all the nodes have been traveled. Then pheromone will be left in the path which an ant has traveled. The shorter the path is, the more pheromone left. Ants can get the optimal solution after a lot of ants have traveled. As the solution procedure of ACO is similar to the design procedure of assembly line, ACO can have a good performance among the meta-heuristic algorithms.

Figure 3 shows the main process of the proposed modified ant colony optimization algorithm. First, we should create a set of alternative nodes according to the precedence constraints and calculate heuristic information and transition probability of all the nodes in the set of alternative nodes. Second, we use roulette wheel selection (Holland 1975) to choose a task. Third, repeat the two steps above

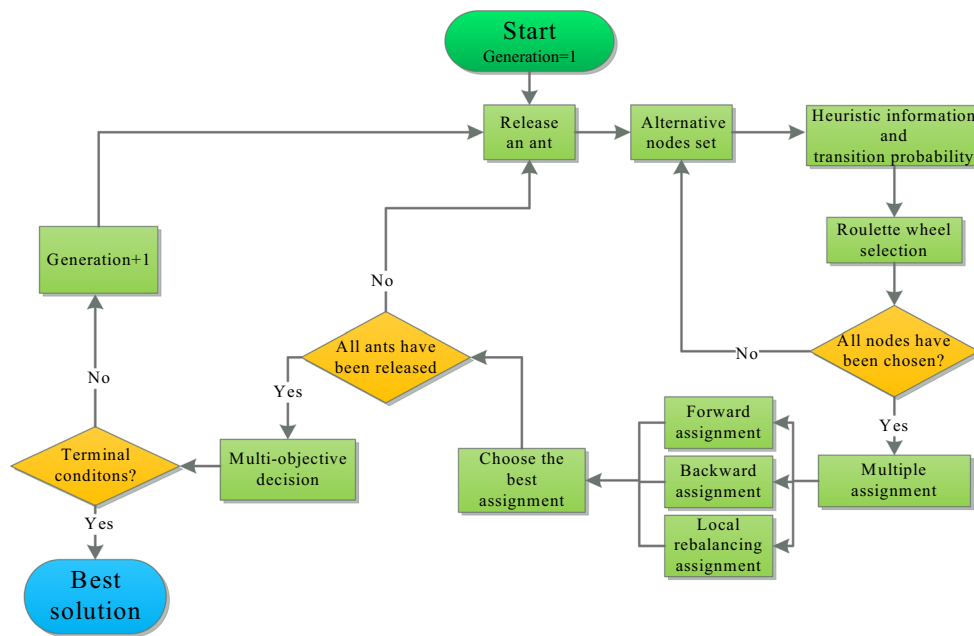


Fig. 3 Flow diagram of the proposed MACO

until all the nodes have been chosen and we can get a task sequence. Fourth, assign task sequence to workstations by forward, backward and local rebalancing assignment approach simultaneously. Forward assignment is to assign task sequence from the first task to the last one to workstations one by one. Backward assignment is to assign task sequence from the last task to the first one to workstations one by one. Local rebalancing assignment is to reassign the workstations between the longest and the shortest workstation which have been assigned by forward assignment. The specific of the three assignment approaches have been introduced in Sect. 4.3. Then calculate objective function of the three assignment approaches, respectively, and choose the best as the assignment approach of this task sequence. Fifth, retain optimal ants using stratified sequential algorithm combined with Pareto-optimal front after all the ants have completed the trip. Sixth, update the pheromone combining with local and global update. Use all the solution in the set of Pareto to update global pheromone trails. Set a limit restrict the quantity of pheromone on each trail. Finally, repeat all the steps above until no better solution in three hundred generations or reach the maximum number of iteration.

4.1 Heuristic information

Ant colony optimization algorithm has some advantages compared with genetic algorithm and particle swarm algorithm. GA and PSO only use heuristic information to generate an initial feasible solution at the beginning of algorithm or

even not. Then the two algorithms only rely on optimization mechanism of themselves to search for optimal solution. However, ACO can take full advantage of heuristic information. ACO utilizes heuristic information each time when choosing nodes to establish a task sequence.

Heuristic information built according to the characteristics of problem can guide an ant to build a feasible solution efficiently and quickly. Heuristic information can affect the quality of the optimal solution which the algorithm searches for. If the heuristic information that we build is not good enough, the direction of the algorithm convergence will be wrong and algorithm could not find good solution.

In this paper, we use subsequent task number  $U_i$  and deviation time  $d_i$  as heuristic information shown in Eq. (8) according to the characteristics of SALBP and objective functions.

$$\eta_i = \omega_1 \cdot \frac{U_i}{\max U_j} + \omega_2 \cdot \frac{1}{d_i + 1}, \quad i, j \in CA \tag{8}$$

$\eta_i$  is heuristic information which is the expectation value of the task  $i$ . Subsequent task number  $U_i$  means the number of the tasks which must be operated later than task  $i$ . If  $U_i$  is big, a lot of tasks can be alternative when task  $i$  has been operated. The bigger  $U_i$  is, the more important task  $i$  is.  $d_i$  represents the deviation time which is the difference between the total time of current workstation and the average time of all the previous workstations when choose task  $i$  and assign it to the current workstation. CA represents the set of alternative tasks.  $\omega_1$  and  $\omega_2$  are the weights of subsequent task number



and deviation time in heuristic information. We suppose that  $\omega_1 = 1$ ,  $\omega_2 = C/5$  after lots of experiments.

Equation (9) shows deviation time  $d_i$ . It means that if we choose task  $i$  and use forward assignment approach to assign task  $i$  to the  $k$ th workstation. Deviation time  $d_i$  is the difference between the total work time of current  $k$ th workstation and the average of the total work time of previous  $k - 1$  workstations. When  $k$ th workstation is the first workstation, deviation time  $d_i$  is equal to 0. As deviation time  $d_i$  can be zero, we might as well set  $d_{i+1}$  as denominator in the heuristic information. We can know that if all workstations have similar work time, the value of line efficiency and workload variation will be good. If all workstations' work time is the same, the value of line efficiency and workload variation will be the optimal value 100% and 0, respectively. If deviation time  $d_i$  is small, it means that the work time of current workstation is close to that of previous workstations. The smaller  $d_i$  is, the higher choice possibility of task  $i$  is.

$$d_i = \begin{cases} \left| T(S_k) - \frac{\sum_{m=1}^{k-1} T(S_m)}{k-1} \right|, & k > 1; \\ 0, & k = 1. \end{cases} \quad (9)$$

Data analysis plays an irreplaceable role during the period of solving a problem. The essence of cognitive analysis is to determine the consistency of features with the expectations of the problems (Ogiela and Ogiela 2009). The most important is to take into consideration both the information of current situation and the impact to further development (Ogiela and Ogiela 2014). It can get twice result with half the effort for finding the core information of the problem. The heuristic information presented in this paper is just in conformity with the point of cognitive analysis that deviation time shows the information of current situation and subsequent task number represents the impact to further development, respectively.

## 4.2 Construction of a feasible solution

### 4.2.1 Set up alternative node set CA

We must set up an alternative node set according to the precedence constraints each time when choosing next node and choose one node from the alternative node set by roulette wheel selection. Only the nodes whose the immediate predecessor tasks have been done can be put into the alternative node set.

In this paper, we use  $N \times N$  matrix Pre to represent the precedence constraints. If task  $i$  is the immediate predecessor task of task  $j$ ,  $\text{Pre}(i, j) = 1$ . If not,  $\text{Pre}(i, j) = 0$ . Thus, we can build a 0–1 matrix of precedence constraints. Equation (10) shows a precedence matrix according to Bowman's precedence graph shown in Fig. 1. The number of tasks  $N$  is 8.

$$\text{Pre} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (10)$$

If  $\sum_{i=1}^N \text{Pre}(i, j) = 0$ , i.e., all the elements of the  $j$ th column of Pre matrix are zero, it means that task  $j$  has no immediate predecessor task, and it can be put into alternative node set. We can know that  $\sum_{i=1}^N \text{Pre}(i, 1) = 0$  from Eq. (10), i.e., task 1 is alternative and it is obvious that only task 1 has no immediate predecessor task at first from Fig. 1.

Suppose that we have chosen task  $i$  and put it into Tabu set which reserve the tasks that have been chosen. Then we must update Pre matrix. Make all the elements of the  $i$ th row of Pre matrix to be zero, i.e.,  $\text{Pre}(i, j) = 0$ ,  $j = 1, 2, \dots, N$ . So that we can choose a task using the method proposed in previous paragraph all the time. If a node has been put into Tabu set, do not put it into alternative node set even though all immediate predecessor tasks of it have been done.

### 4.2.2 Calculate transition probability and select

We must calculate transition probability of all the nodes in alternative node set every time when we are choosing next node and then use roulette wheel selection method to choose a node.

Transition probability  $P_{ij}$  is calculated like Eq. (11).  $i$  is the number of the node which is chosen just last time.  $\tau_{ij}$  represents the amount of pheromone between task  $i$  and task  $j$ .  $\eta_j$  represents the expectation value of task  $j$ .  $\alpha$  and  $\beta$  represent the weight of pheromone and expectation, respectively.

$$P_{ij} = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_j^\beta}{\sum_{l \in \text{CA}} \tau_{il}^\alpha \cdot \eta_l^\beta}, & j \in \text{CA} \\ 0, & j \notin \text{CA} \end{cases} \quad (11)$$

After all the alternative nodes' transition probability have been calculated, we use roulette wheel selection method (Holland 1975) to choose next node. The specific procedures of roulette wheel selection are as follow:

1. Put all the alternative nodes between 0 and 1 one by one.
2. Size of the space that each node occupies is equal to the node's transition probability.
3. Random generate a number between 0 and 1.
4. Select the node whose region contain the generated number.

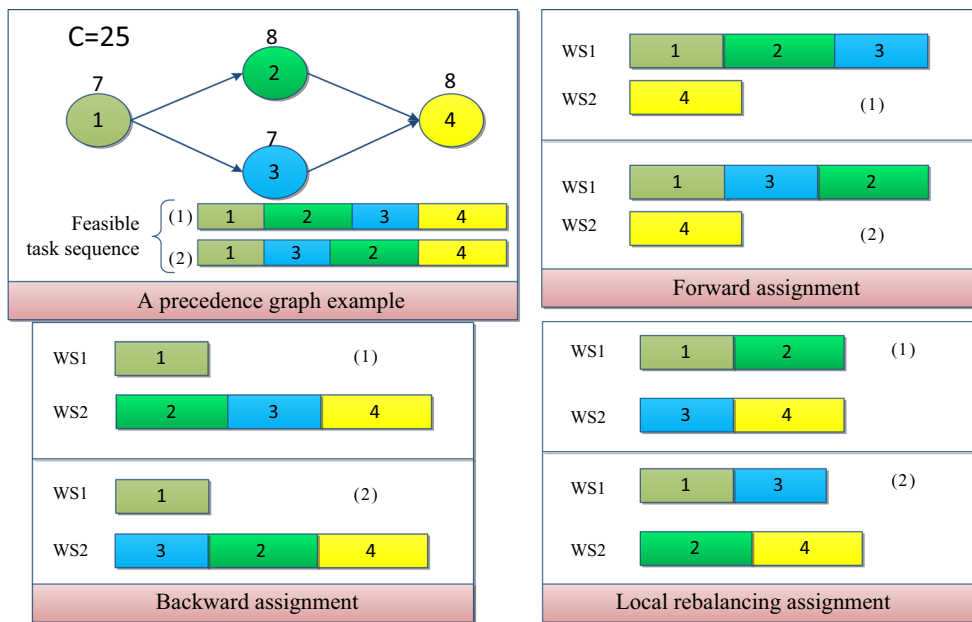


Fig. 4 Multiple assignment illustration

The higher the transition probability of a node is, the higher probability it can be selected. But roulette wheel selection would not neglect the nodes whose transition probability is very small. So that ants can avoid local optimum.

### 4.3 Task assignment

In this paper, we adopt three methods: forward assignment, backward assignment and local rebalancing assignment which have not been used along with ACO before to assign task sequence at the same time. A task sequence assigned by three assignment methods may obtain different results. If we only use forward assignment, we will ignore the other two results which may be better than the result of forward assignment. Figure 4 shows the advantage of using three assignment methods. The first part of Fig. 4 shows a precedence graph for example with a given cycle time 25 and two feasible task sequences of the proposed example. The two feasible task sequences are assigned into workstations by using forward assignment, backward assignment and local rebalancing assignment, respectively, as shown in the rest three parts of Fig. 4. Next, we will specify the three assignment methods.

#### 4.3.1 Forward assignment

In the most of previous papers, they only use forward assignment to assign task sequence. In this paper, when we construct a task sequence, we also only use forward assignment. But after a task sequence has been constructed, we use the other two assignment methods to find if there are better

results. Forward assignment is to assign task sequence from the first task to the last one to the workstation. If a workstation's work time is bigger than cycle time, assign the task to the next workstation until all tasks have been assigned.

#### 4.3.2 Backward assignment

Backward assignment is to assign task sequence from the last task to the first on to the workstation. If a workstations' work time is bigger than cycle time, assign the task to the next workstation until all tasks have been assigned. Then, we must reverse the number of workstations, i.e., the first workstation turns to the last one, the second turns to the penultimate one. So that it can conform to precedence constraint.

#### 4.3.3 Local rebalancing assignment

Local rebalancing assignment is to rebalance some tasks which have been assigned into workstations based on forward assignment. The definite procedures are as follow:

1. After all tasks have been assigned by forward assignment, we can pick out the workstations whose work time is the longest or the shortest. If the number of the workstation which has the longest work time is smaller than that of the workstation which has the shortest work time, go to the second step. If not, go to the third step.
2. If the number of the workstation which has the longest work time is smaller than that of the workstation which has the shortest work time, reallocate tasks in the work-

stations between the longest workstation and the shortest workstation.

3. Calculate the average time of these workstations as first. Then assign these tasks to the workstations one by one. If the difference between a workstation's work time and the average time after a task assigned to this workstation is bigger than that before a task assigned to this workstation, assign this task to the next workstation.
4. If the number of the longest workstation is bigger than that of the shortest workstation, reallocate the tasks in the workstations between the first workstation and the shortest workstation, and the tasks in the workstations between the longest workstation and the last workstation, respectively. The regulation of the reallocation is just like the third step.

In this paper, we simultaneously adopt the three assignment methods and calculate objective function and then choose the best one for the ant. But there are three objectives, and we can not distinguish which is the best one. We use Eq. (12) to evaluate the objective value of the three assignment methods.

$$\begin{aligned}
 F_{v1} &= \frac{\max f_1 - f_1}{\max f_1 - \min f_1} \\
 F_{v2} &= \frac{f_2 - \min f_2}{\max f_2 - \min f_2} \\
 F_{v3} &= \frac{\max f_3 - f_3}{\max f_3 - \min f_3} \\
 F_v &= F_{v1} + F_{v2} + F_{v3}
 \end{aligned} \quad (12)$$

Max  $f_i$  and min  $f_i$ , ( $i = 1, 2, 3$ ) represent the maximum and the minimum of the  $i$ th objective during the three assignment methods.  $F_v$  is the value of evaluation. The bigger of  $F_v$  is, the better the assignment method is.

#### 4.4 Multi-objective decision

In this paper, we combine stratified sequential algorithm with Pareto-optimal front to screen the best ant. We set minimization of the number of workstations as the first level, maximization of line efficiency and minimization of workload variation as the second level. Then we search for Pareto-optimal front in the second level.

##### 4.4.1 Stratified sequential algorithm

Stratified sequential algorithm's basic idea is to divide objective functions into different levels according to the significance of each objective and optimize the latter level objective on the premise that the front level objective has obtained the best.

In this paper, we establish three functions to evaluate the performance of assembly line balancing. We can know that the first objective that minimize the number of workstations is the most significant, and the other two objectives is just to balance the work time of each workstation on the premise of the first objective. So we distribute the first objective that minimize the number of workstations to the first level and the other two objectives to the second level.

##### 4.4.2 Construction of Pareto-optimal front set

Suppose that  $X_1$  and  $X_2$  are any two of feasible solutions. If they can satisfy Eq. (13), it means that  $X_1$  dominate  $X_2$ .

$$\begin{aligned}
 \forall u \in \{1, 2, \dots, U\}, \quad f_u(X_1) &\leq f_u(X_2); \\
 \exists i \in \{1, 2, \dots, U\}, \quad f_i(X_1) &\leq f_i(X_2).
 \end{aligned} \quad (13)$$

If  $X_1$  dominate  $X_2$ , it means that  $X_1$  is better than  $X_2$  on all the objectives. As a consequence, the solution which cannot be dominated by any other solutions can be put into Pareto-optimal front set.

#### 4.5 Pheromone updating rule

In this paper, we update pheromone among the nodes and set a limitation to limit pheromone between  $\tau_{\max}$  and  $\tau_{\min}$ . We use the solutions of each generation for local update and all the solutions in Pareto-optimal front set for global update.

We should calculate the new pheromone  $\Delta\tau_{ij}$  left by ants as Eq. (14) at first.

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \frac{Q}{L} \quad (14)$$

$L$  represents the evaluation of an ant, and it is calculated as Eq. (15).

$$L = \frac{f_1}{\min f_1} - \frac{f_2}{\max f_2} + \frac{f_3}{\min f_3} \quad (15)$$

Min  $f_1$ , max  $f_2$ , min  $f_3$  represent the optimal value of the three objectives in Pareto-optimal front at that time.

After calculating the new left pheromone  $\Delta\tau_{ij}$ , we will update the pheromone among the nodes as Eq. (16).

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij}, \quad (i, j = 1, 2, \dots, N) \quad (16)$$

If some pheromone oversteps the limitation, make it equal to corresponding threshold. Limitation of pheromone can limit the difference among the pheromone on each road to be oversize. As a consequence, it can effectively avoid ants trapping in local optimum.



### 5 Computational experiments and results

In this section, we use the proposed MACO to solve some famous test problems (Scholl 1993; Scholl and Klein 2007) with some different cycle times in order to test its performance for multi-objective SALBP, and compare with MOGA (Hwang et al. 2008) and MA-GA (Al-Hawari et al. 2015).

#### 5.1 Algorithmic and experimental configurations

The experiments are coded on Visual Basic 6.0 and performed on a Core-i3 CPU (2.10 GHZ speed). The parameters of the proposed MACO are shown in Table 1. Terminal conditions of the experiments are shown as follow:

**Table 1** Parameters configurations

Parameter	Value
Population size: <i>PopSize</i>	$2/3 \times N$
Pheromone weight: $\alpha$	1
Heuristic information weight: $\beta$	3
Pheromone volatilization coefficient: $\rho$	0.4
Pheromone concentration: $Q$	40
Pheromone upper limit: $\tau_{max}$	20
Pheromone lower limit: $\tau_{min}$	80

- The number of generations reaches one thousand.
- No better solution appears in three hundred generations.

#### 5.2 Results

We can see the results of the proposed MACO for multi-objective SALBP comparing with MOGA and MA-GA from Table 2. The column of  $m^*$  represents the minimum number of workstations which has been known as the optimal solution. The results shown in bold fonts are better than that of both MOGA and MA-GA. The last column shows the assignment methods that the optimal solution chooses.

Figures 5 and 6 show the values of line efficiency and workload variation of all the algorithms using histogram. We can see the difference of the performance of the three algorithms for ALBP easily. Figures 7 and 8 show the improvement of results that the proposed MACO obtains comparing with MOGA and MA-GA.

Figure 9 shows the convergence line chart of workload variation obtained from the 18th test problem. We can see that the proposed MACO can get better solution at the beginning of iteration. The main reason is that the suitable heuristic information plays a leading role during ants search for solutions.

Table 3 shows the results of statistical treatment that each problem has been run for 30 times by the proposed MACO.

**Table 2** Experimental results of the proposed MACO versus MOGA and MA-GA

No.	Test problem	<i>N</i>	<i>C</i>	$m^*$	MOGA			MA-GA			Proposed MACO			Assignment <sup>a</sup>
					<i>m</i>	<i>E%</i>	<i>W<sub>v</sub></i>	<i>m</i>	<i>E%</i>	<i>W<sub>v</sub></i>	<i>m</i>	<i>E%</i>	<i>W<sub>v</sub></i>	
1	Mitchell	21	14	8	8	93.7	0.042	8	93.7	0.042	8	93.7	0.042	1, 2, 3
2			15	8	8	87.5	0.090	8	93.8	0.042	8	93.8	0.042	1, 3
3			21	5	5	100.0	0.000	5	100.0	0.000	5	100.0	0.000	1, 2, 3
4	Heskia	28	138	8	8	92.7	0.022	8	99.2	0.006	8	99.2	0.006	1
5			205	5	5	99.9	0.001	5	99.9	0.001	5	99.9	0.001	1, 2, 3
6			324	4	4	79.0	0.153	4	100.0	0.000	4	100.0	0.000	3
7	Sawyer	30	27	13	13	92.3	0.035	13	95.9	0.032	13	95.9	<b>0.024</b>	1, 3
8			33	11	11	89.2	0.046	11	92.1	0.036	11	92.1	<b>0.034</b>	1
9			54	7	7	85.7	0.030	7	96.4	0.024	7	96.4	<b>0.015</b>	1, 3
10	Kilbridge	45	79	7	7	99.8	0.004	7	99.8	0.004	7	99.8	0.004	1, 2, 3
11			92	6	6	100.0	0.000	6	100.0	0.000	6	100.0	0.000	1, 2, 3
12			184	3	3	100.0	0.000	3	100.0	0.000	3	100.0	0.000	1, 2, 3
13	Tonge	70	364	10	10	96.4	0.039	10	98.6	0.007	10	<b>99.4</b>	<b>0.004</b>	1, 3
14			468	8	8	93.8	0.014	8	97.9	0.009	8	<b>99.5</b>	<b>0.003</b>	1
15	Arcus1	83	6842	12	12	92.2	0.043	12	96.9	0.041	12	<b>97.6</b>	0.041	1, 3
16			8412	10	10	90.0	0.038	10	97.1	0.015	10	<b>98.3</b>	<b>0.007</b>	3
17			10,816	8	8	87.5	0.160	8	92.4	0.048	8	<b>97.6</b>	<b>0.011</b>	3
18	Arcus2	111	10,027	16	16	93.7	0.030	16	96.7	0.025	16	<b>99.4</b>	<b>0.004</b>	1
19			10,743	15	15	93.3	0.035	15	98.3	0.014	15	<b>98.7</b>	<b>0.012</b>	3

<sup>a</sup> Assignment-1, forward assignment; 2, backward assignment; 3, local rebalancing assignment

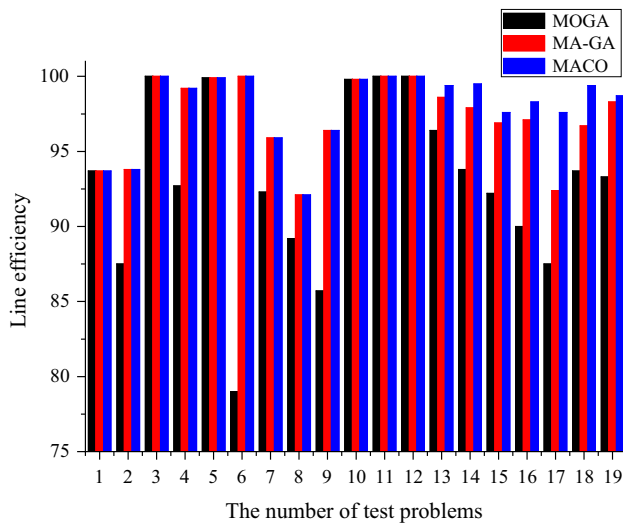


Fig. 5 Line efficiency values of algorithms

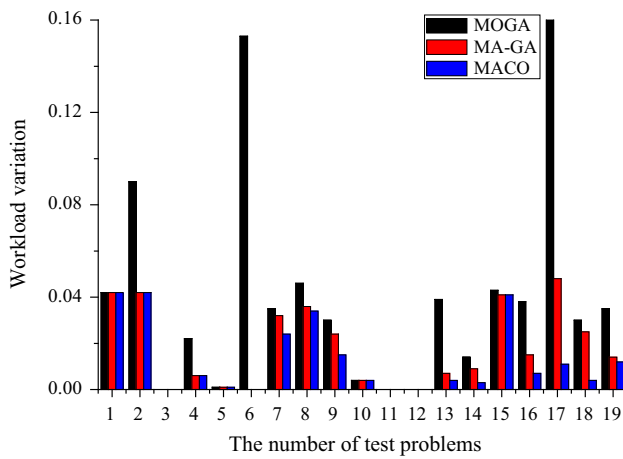


Fig. 6 Workload variation values of algorithms

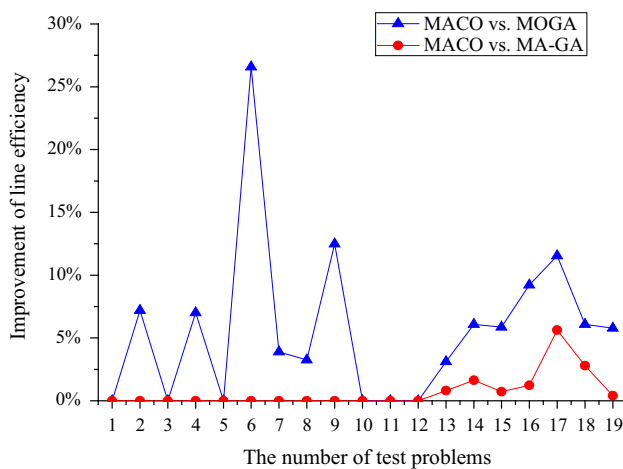


Fig. 7 Improvement of line efficiency

The notation of Best, Worst, Average, SD and ACT existing in the table represent the best value, worst value, average

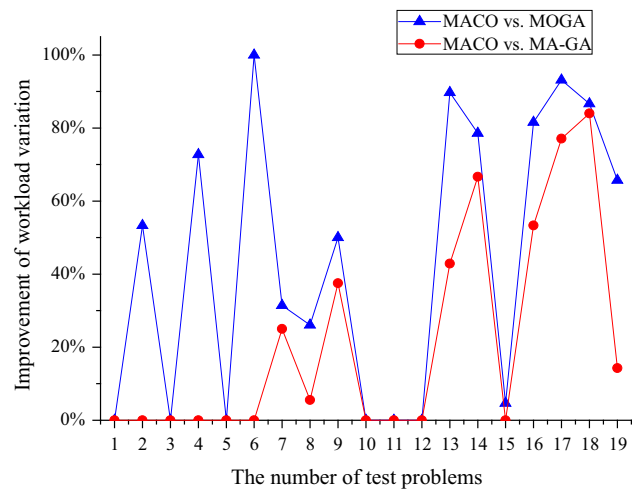


Fig. 8 Improvement of workload variation

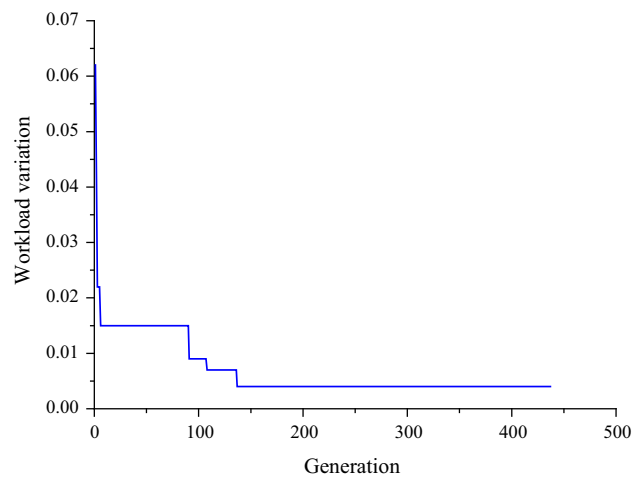


Fig. 9 Convergence line chart of workload variation

value, standard deviation and average convergence time of 30 times results of each problem, respectively.

Figure 10 shows the standard deviation of the 30 times results. Figure 11 shows the tendency of average convergence time as the problem size  $N$  increasing.

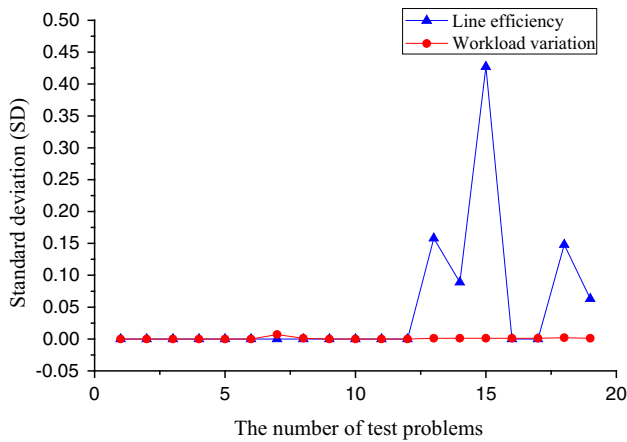
Figures 12 and 13 show the Gantt chart of two problems' optimal solution which the other two algorithms MOGA and MA-GA can not get. The two test problems are Sawyer and Arcus1 whose given cycle time are 27 and 8412, respectively.

### 6 Discussion

It is observed that the proposed MACO performs better than MOGA and MA-GA for all the test problems from Table 2. The proposed MACO can acquire the optimal number of workstations and the same or better line efficiency and workload variation than MOGA and MA-GA. The improvement

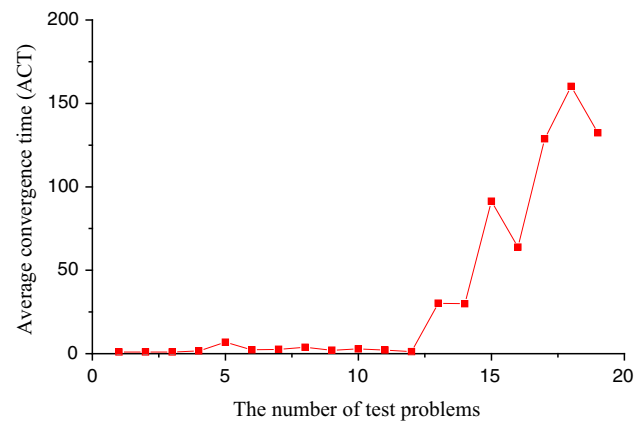
**Table 3** Statistical treatment of the proposed MACO for 30 times results of each problem

No.	Test problem	N	C	m	Best		Worst		Average		SD		ACT (s)
					E%	W <sub>v</sub>	E%	W <sub>v</sub>	E%	W <sub>v</sub>	E%	W <sub>v</sub>	
1	Mitchell	21	14	8	93.7	0.042	93.7	0.042	93.7	0.042	0.000	0.000	1.0
2			15	8	93.8	0.042	93.8	0.042	93.8	0.042	0.000	0.000	1.0
3			21	5	100.0	0.000	100.0	0.000	100.0	0.000	0.000	0.000	0.000
4	Heskia	28	138	8	99.2	0.006	99.2	0.006	99.2	0.006	0.000	0.000	1.6
5			205	5	99.9	0.001	99.9	0.001	99.9	0.001	0.000	0.000	6.8
6			324	4	100.0	0.000	100.0	0.000	100.0	0.000	0.000	0.000	2.3
7	Sawyer	30	27	13	95.9	0.024	95.9	0.043	95.9	0.036	0.000	0.007	2.6
8			33	11	92.1	0.034	92.1	0.036	92.1	0.035	0.000	0.001	3.8
9			54	7	96.4	0.015	96.4	0.015	96.4	0.015	0.000	0.000	2.1
10	Kilbridge	45	79	7	99.8	0.004	99.8	0.004	99.8	0.004	0.000	0.000	2.8
11			92	6	100.0	0.000	100.0	0.000	100.0	0.000	0.000	0.000	2.2
12			184	3	100.0	0.000	100.0	0.000	100.0	0.000	0.000	0.000	1.2
13	Tonge	70	364	10	99.4	0.004	98.9	0.005	99.3	0.004	0.158	0.001	30.2
14			468	8	99.5	0.003	99.3	0.005	99.5	0.003	0.089	0.001	29.9
15	Arcus1	83	6842	12	97.6	0.041	96.5	0.043	97.2	0.042	0.427	0.001	91.3
16			8412	10	98.3	0.007	98.3	0.011	98.3	0.008	0.000	0.001	63.8
17			10,816	8	97.6	0.011	97.6	0.012	97.6	0.011	0.000	0.001	128.8
18	Arcus2	111	10,027	16	99.4	0.004	99.1	0.010	99.3	0.006	0.148	0.002	160.2
19			10,743	15	98.7	0.012	98.6	0.014	98.7	0.013	0.063	0.001	132.4



**Fig. 10** Standard deviation of the proposed MACO

of the proposed MACO can be a maximum of 26.58 and 100% improvement than MOGA for the second and the third objective, respectively, and a maximum of 5.6 and 84% improvement than MA-GA. MACO has good performance for large-scale ALBP such as test problem Arcus2. As a consequence of that, we can say that proposed MACO has excellent performance for both small-scale and large-scale SALBP. Furthermore, we can see that the standard deviation is very small for all the test problem. It demonstrates that the proposed MACO has an excellent stability for all kinds of SALBP. The convergence time of the proposed MACO is



**Fig. 11** Average convergence time of the proposed MACO

also in a reasonable scope no matter for small-size or large-size problems observed from Fig. 11.

The larger the scale of assembly line is, the more feasible solutions exist. In this paper, the proposed MACO can search for the better solutions than MOGA and MA-GA from the large number of feasible solutions. The main reason for this is that we establish a suitable heuristic information. We can see it from Fig. 9 which shows the convergence line of MACO for Arcus2 test problem. The proposed MACO can search for better solutions at the beginning of iteration because we have established a suitable heuristic information. A suitable

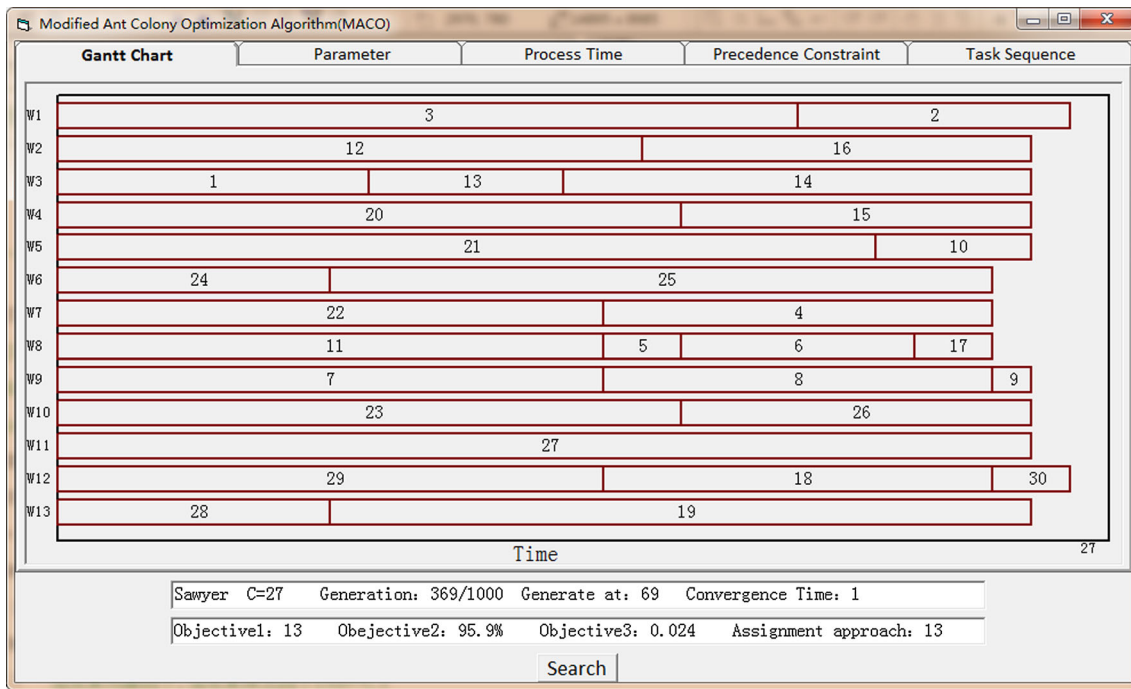


Fig. 12 Optimal solution of Sawyer test problem for which the given C is 27

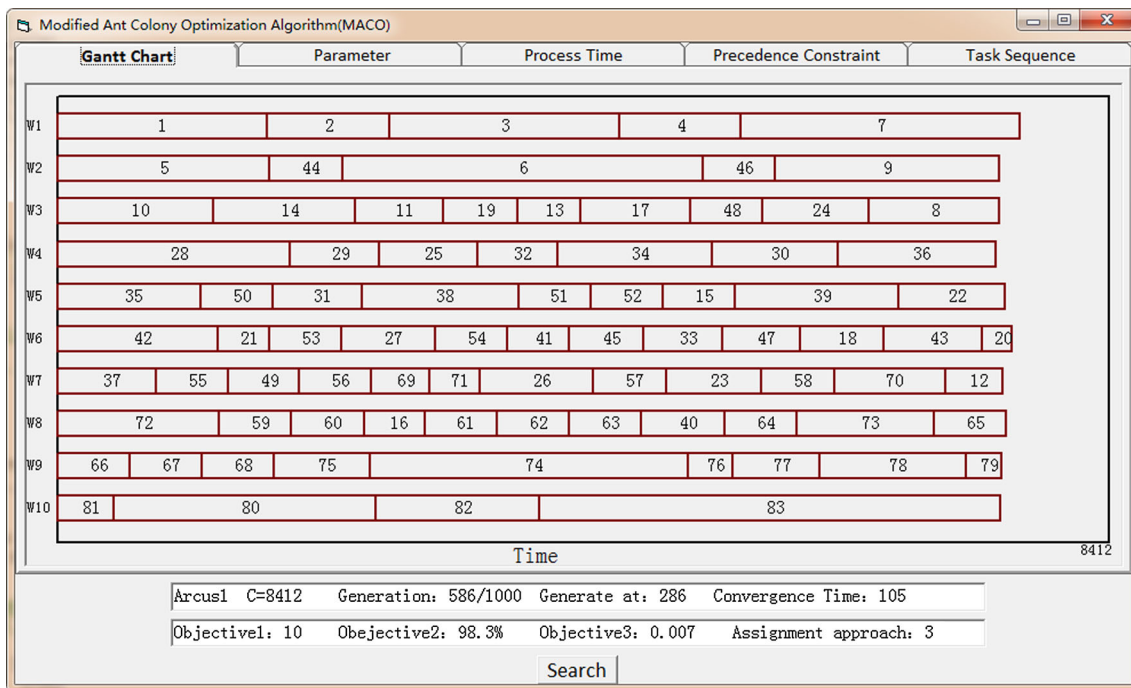


Fig. 13 Optimal solution of Arcus1 test problem for which the given C is 8412

heuristic information can actively guide ants to search for the optimal solution accurately and rapidly. It is different from pheromone which impacts ants by feedback derived from optimization mechanism of ACO itself. Pheromone can only passively affect ants. Heuristic information must be established according to objectives. A wrong heuristic information

may guide ants to the wrong direction and could not find the optimal solution forever. In this paper, we establish heuristic information with subsequent task number and deviation time according to objectives of SALBP. It proves that the proposed heuristic information has a good performance on SALBP according to the test results in Sect. 5. Such as the tests of

nos. 7, 8, 9 and so on which can get better solutions than MOGA and MA-GA using forward assignment. These tests which eliminate the influence of multiple assignments fully prove that the proposed heuristic information has good effect.

Besides that, the proposed multiple assignment approach and multi-objective decision that stratified sequential algorithm combined with Pareto-optimal front also play indispensable roles. Multiple assignment approach can assign a task sequence to different workstations and get different objective value as a result. So that it can increase the potential of a task sequence. As the proposed heuristic information tends to assign a task whose processed time is short to the bottom of a workstation, it is beneficial for local rebalancing assignment to reallocate the short tasks at the bottom of a workstation to the next workstation to seek better solutions and does not destroy the cycle constraint. We can see that the 6th, 16th, 17th and 19th test problems in Table 2 can be searched for the optimal solution only by local rebalancing assignment method. It proves that multiple assignment approach plays a great role in MACO. The proposed multi-objective decision which combines stratified sequential algorithm with Pareto-optimal front is able to screens much more and better solutions than linear weighted sum method. The more and better solutions screened for pheromone global update can improve the influence of pheromone on the Pareto-optimal solutions' path and guide ants to better path.

## 7 Conclusions

The main contribution of this paper is the improvement of a modified ant colony optimization algorithm for multi-objective SALBP using the novel heuristic information, multiple assignment method and a modified multi-objective decision. The proposed MACO has a good performance on multi-objective SALBP which is to minimize the number of workstations, maximize line efficiency and minimize workload variation as a consequence of test problem experiments. It is obvious that the proposed MACO has improved results compared to MOGA and MA-GA. The maximum improvement of the proposed MACO can be 26.58 and 100 % improvement than MOGA and 5.6 and 84 % improvement than MA-GA on the second objective and the third objective, respectively. The proposed MACO also has an excellent stability as the standard deviation is always less than 0.427 for all the test problems. The computing speed of the proposed MACO is also very fast. It can get optimal solution within 200 s even though the test problem is Arcus1 whose problem size is reach up to 111 tasks.

In the future, we will continue to study ALBP with respect to other objective functions and improve ACO for the new objectives. We will also investigate to combine ACO with

other meta-heuristic algorithm for ALBP. The contribution of this paper put forward a new idea to improve ACO according to the characteristic of the problem to be solved.

**Acknowledgements** This work is supported by the National Natural Science Foundation of China (No. 51275104) and supported in part by the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20132304120021).

### Compliance with ethical standards

**Conflict of interest** The authors Yu-guang Zhong and Bo Ai declare that they have no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Informed consent** Informed consent was obtained from all individual participants included in the study.

## References

- Al-Hawari T, Ali M, Al-Araidah O, Mumani A (2015) Development of a genetic algorithm for multi-objective assembly line balancing using multiple assignment approach. *Int J Adv Manuf Technol* 77:1419–1432. doi:[10.1007/s00170-014-6545-5](https://doi.org/10.1007/s00170-014-6545-5)
- Bautista J, Pereira J (2007) Ant algorithms for a time and space constrained assembly line balancing problem. *Eur J Oper Res* 177:2016–2032
- Baybars I (1986) A survey of exact algorithms for the simple assembly line balancing problem. *Manag Sci* 32:909–932
- Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. *Eur J Oper Res* 168:694–715. doi:[10.1016/j.ejor.2004.07.023](https://doi.org/10.1016/j.ejor.2004.07.023)
- Bowman EH (1960) Assembly-line balancing by linear programming. *Oper Res* 8:385–389
- Boysen N, Fliedner M, Scholl A (2007) A classification of assembly line balancing problems. *Eur J Oper Res* 183:674–693
- Dou J, Li J, Su C (2013) A novel feasible task sequence-oriented discrete particle swarm algorithm for simple assembly line balancing problem of type 1. *Int J Adv Manuf Technol* 69:2445–2457. doi:[10.1007/s00170-013-5216-2](https://doi.org/10.1007/s00170-013-5216-2)
- Gutiérrez C, García-Magariño I (2011) Modular design of a hybrid genetic algorithm for a flexible job-shop scheduling problem. *Knowl Based Syst* 24:102–112
- Hamta N, Ghomi SMTF, Jolai F, Shirazi MA (2013) A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *Int J Prod Econ* 141:99–111
- Held M, Karp RM, Shreshian R (1963) Assembly-line balancing-dynamic programming with precedence constraints. *Oper Res* 11:442–459
- Holland JH (1975) *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor
- Hou L, Wu YM, Lai RS, Tsai CT (2014) Product family assembly line balancing based on an improved genetic algorithm. *Int J Adv Manuf Technol* 70:1775–1786
- Hwang RK, Katayama H, Gen M (2008) U-shaped assembly line balancing problem with genetic algorithm. *Int J Prod Res* 46:4637–4649. doi:[10.1080/00207540701247906](https://doi.org/10.1080/00207540701247906)
- Jackson JR (1956) A computing procedure for a line balancing problem. *Manag Sci* 2:261–271



- Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer applications. Plenum Press, New York, pp 85–104
- Kilinc O (2011) Firing sequences backward algorithm for simple assembly line balancing problem of type 1. *Comput Ind Eng* 60:830–839. doi:[10.1016/j.cie.2011.02.001](https://doi.org/10.1016/j.cie.2011.02.001)
- Li D, Zhang C, Shao X, Lin W (2014) A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. *J Intell Manuf*. doi:[10.1007/s10845-014-0919-2](https://doi.org/10.1007/s10845-014-0919-2)
- Lv Q (2011) Simple assembly line balancing using particle swarm optimization algorithm. *Int J Digit Content Technol Appl* 5:297–304. doi:[10.4156/jdcta.vol5.issue6.36](https://doi.org/10.4156/jdcta.vol5.issue6.36)
- Ogiela L, Ogiela MR (2009) Cognitive techniques in visual data interpretation. In: *Studies in computational intelligence*, vol 228. Springer, Berlin, Heidelberg
- Ogiela L, Ogiela MR (2014) Cognitive systems for intelligent business information management in cognitive economy. *Int J Inf Manag* 34:751–760
- Petropoulos DI, Nearchou AC (2011) A particle swarm optimization algorithm for balancing assembly lines. *Assem Autom* 31: 118–129
- Salveson ME (1955) The assembly line balancing problem. *J Ind Eng* 6:18–25
- Scholl A (1993) Data of assembly line balancing problems. Publications of Darmstadt Technical University Institute for Business Studies
- Scholl A, Klein R (2007) Assembly line balancing. <http://alb.mansci.de/>
- Sfrent A, Pop F (2015) Asymptotic scheduling for many task computing in big data platforms. *Inf Sci* 319:71–91
- Vasile MA, Pop F, Tutueanu RI, Cristea V, Kolodziej J (2015) Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. *Future Gener Comput Syst* 51:61–71
- Yagmahan B (2011) Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. *Expert Syst Appl* 38:12453–12461. doi:[10.1016/j.eswa.2011.04.026](https://doi.org/10.1016/j.eswa.2011.04.026)
- Yu J, Yin Y (2010) Assembly line balancing based on an adaptive genetic algorithm. *Int J Adv Manuf Technol* 48:347–354. doi:[10.1007/s00170-009-2281-7](https://doi.org/10.1007/s00170-009-2281-7)