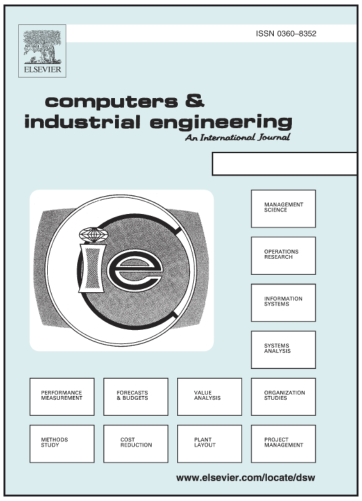


New MILP model and station-oriented ant colony optimization algorithm for balancing U-type assembly lines

Zixiang Li, Ibrahim Kucukkoc, Qiuhua Tang

PII: S0360-8352(17)30295-4
DOI: http://dx.doi.org/10.1016/j.cie.2017.07.005
Reference: CAIE 4812

To appear in: *Computers & Industrial Engineering*

Received Date: 22 January 2017
Revised Date: 3 July 2017
Accepted Date: 5 July 2017

New MILP model and station-oriented ant colony optimization algorithm for balancing U-type assembly lines

Zixiang Li [a,b], Ibrahim Kucukkoc [c*], Qiuhua Tang [a,b]

[a] Key Laboratory of Metallurgical Equipment and Control Technology, Wuhan University of Science and Technology, Wuhan, Hubei, China.
[b] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, Hubei, China.
Email: zixiangliwust@gmail.com (Z. Li); tangqiuhua@wust.edu.cn (Q. Tang)
[c] Industrial Engineering Department, Balikesir University, Cagis Campus, Balikesir 10145, Turkey.
Email: i.kucukkoc@exeter.ac.uk

*Corresponding author

Abstract: U-type assembly lines are extensively applied in modern manufacturing systems for higher flexibility and productivity. This research presents a new mixed-integer linear programming model to minimize the number of stations, where one expression is used to represent the precedence relationship constraint rather than two expressions as in published researches. The proposed model is compared to three other models and the correctness or the incorrectness of these models are analyzed by enumerating all possible allocations between the two tasks. The comparison makes it clear that the proposed model iterates fast and achieves competing results. Additionally, a modified ant colony optimization approach, referred to as station-oriented ant colony optimization algorithm, is proposed to tackle large-size problems. This method generates a set of task assignments and selects the best one for the current station, rather than obtaining only one task assignment at a time. A set of benchmark problems is solved using the proposed method and the results are compared to those obtained by the state-of-the-art methods (including ULINO) and the variants of ant colony optimization approach. The computational study demonstrates the superiority of the proposed method over the compared ones as it achieves optimal solutions for 255 cases (out of 269) and outperforms the current best method, ULINO, for 21 cases. It is also worthy to mention that the station-oriented procedure improves the performance of original ant colony optimization by a significant margin.

Keywords: Assembly line balancing; U-type assembly line; Integer programming; Ant colony optimization; Artificial intelligence

New MILP model and station-oriented ant colony optimization algorithm for balancing U-type assembly lines

Abstract: U-type assembly lines are extensively applied in modern manufacturing systems for higher flexibility and productivity. This research presents a new mixed-integer linear programming model to minimize the number of stations, where one expression is used to represent the precedence relationship constraint rather than two expressions as in published researches. The proposed model is compared to three other models and the correctness or the incorrectness of these models are analyzed by enumerating all possible allocations between the two tasks. The comparison makes it clear that the proposed model iterates fast and achieves competing results. Additionally, a modified ant colony optimization approach, referred to as station-oriented ant colony optimization algorithm, is proposed to tackle large-size problems. This method generates a set of task assignments and selects the best one for the current station, rather than obtaining only one task assignment at a time. A set of benchmark problems is solved using the proposed method and the results are compared to those obtained by the state-of-the-art methods (including ULINO) and the variants of ant colony optimization approach. The computational study demonstrates the superiority of the proposed method over the compared ones as it achieves optimal solutions for 255 cases (out of 269) and outperforms the current best method, ULINO, for 21 cases. It is also worthy to mention that the station-oriented procedure improves the performance of original ant colony optimization by a significant margin.

Keywords: Assembly line balancing; U-type assembly line; Integer programming; Ant colony optimization; Artificial intelligence

1. Introduction

In conjunction with the industrial revolution emerged in the 18th century, mass production techniques have gained enormous importance. Assembly lines, one of the most efficient of those mass production techniques, have been put into practice by Henry Ford and his colleagues pioneering the advances in producing homogeneous products in mass quantities. As the flexibility has become one of the crucial factors which affect the survivability of companies in parallel to changes in consumer-centric market, assembly lines faced changes in terms of configuration (I. Kucukkoc & Zhang, 2016).

An assembly line is a sequence of workstations (or stations) linked to each other via a conveyor or moving belt on which products are transported from one workstation to another. Some tasks are performed in each station by workers located in this station within the time allowed, called cycle time. The problem of deciding on which task will be conducted in which station is called the assembly line balancing problem and characterized based on various factors. Some of those are the configuration of the line, the variety of products assembled on the line and the allocation of workers or stations across the line. In its traditional version, a simple and straight assembly line, which consists of serially linked stations to perform tasks with the aim of producing a complete product, was assumed by Salveson (1955). However, about four decades later, Miltenburg and Wijngaard (1994) introduced the U-shaped assembly lines which provide more efficiency as well as flexibility. As seen in Fig.1, the line constitutes a U-shape in such a way that the entrance and the exit of the line are located close to each other. Thus, workers operating located on one branch of the line may work on the other branch in the same cycle, which helps increase the labor productivity.

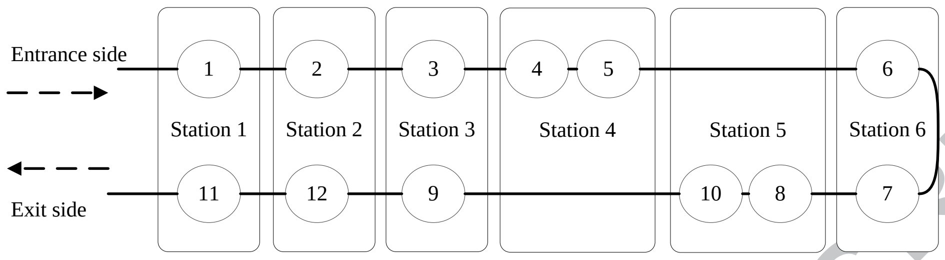


Fig.1 Layout of a U-type assembly line

1.1. Analysis of the literature

Miltenburg and Wijngaard (1994) addressed to the simplest form of the U-shaped assembly line design and discussed its advantages over traditional straight lines. Many experimental and case-based research followed this and verified the efficiency of U-shaped lines over straight lines in different aspects. Battaïa and Dolgui (2013) presented a taxonomy of line balancing problems and Kucukkoc and Zhang (2015) summarized the majority of the research on U-shaped assembly lines, particularly. Urban (1998) and Urban and Chiang (2006) developed mathematical formulations (an integer programming formulation and a chance-constrained piecewise-linear program, respectively) for optimally balancing the single-model U-shaped lines with the aim of minimizing the number of stations. However, as the considered problem is NP-hard, metaheuristics are of good choice due to easy implementation and fast speed (see, for example, Modiri-Delshad et al. (2016), Kaboli et al. (2016), De et al. (2016), Pan et al. (2017), Samà et al. (2017), Aghay Kaboli et al. (2017), Mogale et al. (2017), De et al. (2017), Maiyar and Thakkar (2017) for some recent applications of metaheuristics on various NP-hard optimization problems. Therefore, the balancing effort for solving the U-shaped assembly line balancing problem (UALBP) majorly focused on developing heuristic and metaheuristic algorithms for its variants. The attention paid to the mathematical model was not as high as anticipated. Erel et al. (2001), Hamzadayi and Yildiz (2013) and Manavizadeh et al. (2013) developed simulated annealing approaches while Hwang et al. (2008) and Hamzadayi and Yildiz (2012) proposed genetic algorithm approaches for solving the UALBP with the aim of minimizing the number of stations. Other heuristics/metaheuristics developed on the same problem are as follows; a shortest route formulation by Gökçen et al. (2005), a hybrid heuristic algorithm by Chiang and Urban (2006), a heuristic approach by Toksari et al. (2008), an ant colony optimization (ACO) approach by Sabuncuoglu et al. (2009), and a multi-pass random assignment algorithm by Yegul et al. (2010). Chiang and Urban (2006) assumed stochastic task times and proposed a hybrid heuristic as a balancing solution procedure. Cycle time minimization was also considered by Scholl and Klein (1999), Gökçen and Ağpak (2006), Chiang et al. (2007), Kara et al. (2009) and Rabbani et al. (2012) as an additional objective to the minimization of the number of stations. Scholl and Klein (1999) developed a branch and bound procedure called ULINO and Gökçen and Ağpak (2006) developed a goal programming formulation for single model UALBP. While a binary fuzzy goal programming approach and more than one optimal solution strategy were proposed by Kara et al. (2009) and Chiang et al. (2007), respectively, for traditional U-lines; a genetic algorithm based approach was proposed by Rabbani et al. (2012) for balancing two-sided lines with multiple U-shaped layout. Kucukkoc and Zhang (2015; 2017) introduced parallel U-shaped lines and proposed constructive heuristic algorithms for balancing the lines considering single and mixed-model production.

Kazemi et al. (2011) proposed a two-stage genetic algorithm for mixed-model U-line balancing problem considering duplicated tasks. Aase et al. (2004) conducted an experimental study on U-shaped line design and showed its effect on labor productivity. Kim et al. (2006) and Özcan et al. (2011) proposed an endosymbiotic evolutionary algorithm and a genetic algorithm, respectively, for solving both line balancing and model sequencing problems simultaneously as well as considering the absolute deviation of workloads in a mixed-model production environment. In addition, simulated annealing algorithms were proposed by Kara et al. (2007) and Dong et al. (2014) for solving the same problem.

Although some attention was paid to the mathematical model, the models obtained eventually were far from achieving the state-of-the-art results. Aase et al. (2003) analyzed exact U-shaped line balancing procedures. Fattahi and Turkay (2015) investigated the use of either-or precedence relationship constraints in mathematical formulations given in previous studies and illustrated that using such constraints may cause infeasibility. They revised the MILP model for UALBP and tested its efficiency on benchmark problems. To the best of the authors' knowledge, ULINO published in 1999 remains the best performer though several improvements have been gained in the technology and so the computing facilities have developed since then. On the other hand, there are several mathematical formulations including some wrong ones, but there is no research where all the models are evaluated.

Some other objectives were also sought by researchers. Kara et al. (2011) proposed an integer programming formulation for minimizing total operating costs by considering resource-dependent task times. Celik et al. (2014) developed an ant colony optimization approach with the aim of minimizing total rebalancing cost considering stochastic times. Another cost minimization based research was conducted by Jayaswal and Agarwal (2014). Station and resource utilization costs were aimed to be minimized when considering resource-dependent task times.

1.2. Contributions of the work

This paper presents several contributions as follows. First, a new MILP model is developed to deal with the precedence relationship constraint, followed by a comprehensive analysis of the proposed model and other models provided by Urban (1998), Urban and Chiang (2006), Kazemi et al. (2011), Aase et al. (2004; 2003) and Fattahi and Turkay (2015). The comparative study shows that the proposed model iterates fast and produces competing results when executed via the Cplex solver of GAMS. Second, a station-oriented ant colony optimization approach (called SACO hereafter) is developed for solving large-sized UALBPs for the first time. This new algorithm differs from the original ACO in that the SACO utilizes a new pheromone trail to select the first task and proposes a station-oriented decoding mechanism. This mechanism selects the best set of tasks for the current station from a set of possible assignments. Third, a comprehensive computational study is conducted between the proposed SACO and state-of-the-art methods, including ULINO and ACO variants. A total of 269 test problems (summarized by Scholl and Klein (1999)) are solved for this aim. The results show that SACO is the best performer among those tested as it achieves all the current best results and optimal solutions of 255 cases. Furthermore, for 21 cases, SACO outperforms ULINO, which is the current best performer for UALBP.

The remainder of this paper is organized as follows. Section 2 presents the newly developed mathematical model and analyses its differences from the existing models. Section 3 describes the running mechanism of the proposed SACO algorithm in details and Section 4 reports the results of the comprehensive computational study conducted. Finally, Section 5 concludes the paper by summarizing the main findings and the real-world application opportunities, followed by the directions for future research.

2. Mathematical model

This section introduces the applied models in literature and the proposed model, and carries out correctness analysis by enumerating all the possible allocations between two tasks.

2.1 Developed models

UALBP differs from SALBP in the precedence constraint. A task in SALBP is assignable when all its predecessors have been allocated whereas a task in UALBP is assignable when all its predecessors or successors have been allocated. To be precise, a task on the entrance side is assigned after assignment of all its predecessors and a task on the exit side is assigned after assignment of all its successors.

To formulate the precedence relations, the first applied method employs 0-1 variables to describe either–or scenario (Aase et al. (2004; 2003) and Kazemi et al. (2011)), referred to as Model 1. Model 1, taken from the literature with no change, is as follows:

Minimize $\sum_{j=1}^{m} z_j$ (1)

Subject to

$\sum_{j=1}^{m} a_{ij}=1$ for $i=1,\ldots,n$ (2)

$\sum_{i=1}^{n} t_i \cdot a_{ij} \le \mathrm{CT}$ for $j=1,\ldots,m$ (3)

$\sum_{j=1}^{m} j\cdot\left(a_{pj}-a_{qj}\right) \le \psi\cdot u_q \ \forall (p,q)\in\wp$ (4)

$\sum_{j=1}^{m} j\cdot\left(a_{qj}-a_{pj}\right) \le \psi\cdot\left(1-u_p\right) \ \forall (p,q)\in\wp$ (5)

$\sum_{i=1}^{n} a_{ij} \le \psi\cdot z_j$ for $j=1,\ldots,m$ (6)

$a_{ij},u_i,z_j\in\{0,1\}$ for all i,j (7)

where i, p and q are the indices of tasks, and n is the total number of tasks. Parameter j is the index for stations, and m is the upper bound on the number of stations. z_j is a binary variable to check whether station j is utilized (1, if station j is utilized; 0, otherwise). a_{ij} is a binary variable to describe the task assignment. If task i is allocated to station j, a_{ij}=1; otherwise, a_{ij} =0. Parameter t_i denotes the operation time of task i, CT implies the cycle time and ψ is a very large positive number. Variable u_i is also a binary variable to express either–or constraint, and $\wp=\{(p,q)\}$ is the set of paired tasks, where task p is the immediate predecessor of task q. Equations (1-3) are identical to those for SALBP formulation, and thus this section mainly explains the precedence constraint in Equations (4-5). Equation (4) indicates that task q is assignable when all its predecessors have been allocated, whereas Equation (5) implies that task p is assignable if all its successors have been allocated. The basic logic is that u_i is equal to 0 if task i is allocated to the entrance side, and it is equal to 1 if task i is allocated to the exit side.

Nevertheless, Model 1 might find infeasible solutions (Fattahi & Turkay, 2015) and the main reason lying behind is that Model 1 ignores the situations when two tasks are allocated to entrance side and exit side, respectively. Hence, Fattahi and Turkay (2015) introduced a revised edition, named Model 2 hereafter, where Equations (8-9) are utilized to replace Equations (4-5). The basic logic is that u_i is equal to 1 if task i is allocated to the entrance side, and it is equal to 0 if task i is allocated to the exit side.

$$\sum_{j=1}^{m} j\cdot a_{pj} - \sum_{j=1}^{m} j\cdot a_{qj} \leq m\cdot\left(1+u_p-2\cdot u_q\right)\ \forall(p,q)\in\wp \tag{8}$$

$$\sum_{j=1}^{m} j\cdot a_{qj} - \sum_{j=1}^{m} j\cdot a_{pj} \leq m\cdot u_p\ \forall(p,q)\in\wp \tag{9}$$

Another mathematical model for UALBP is developed by Urban (1998) and Urban and Chiang (2006), referred to as Model 3. Model 3 is formulated as follows with small adjustments, where x_{ij} and y_{ij} are the binary variables to describe the task assignment. If task i is allocated to the entrance side of station j, x_{ij}=1; otherwise, x_{ij}=0. If task i is allocated to the exit side of station j, y_{ij}=1; otherwise, y_{ij}=0.

$$\text{Minimize } \sum_{j=1}^{m} z_j \tag{10}$$

Subject to

$$\sum_{j=1}^{m}\left(x_{ij}+y_{ij}\right)=1 \text{ for } i=1,\ldots,n \tag{11}$$

$$\sum_{i=1}^{n} t_i\cdot\left(x_{ij}+y_{ij}\right)\leq \mathrm{CT} \text{ for } j=1,\ldots,m \tag{12}$$

$$\sum_{j=1}^{m}\left(m-j+1\right)\cdot\left(x_{pj}-x_{qj}\right)\geq 0\ \forall(p,q)\in\wp \tag{13}$$

$$\sum_{j=1}^{m}\left(m-j+1\right)\cdot\left(y_{qj}-y_{pj}\right)\geq 0\ \forall(p,q)\in\wp \tag{14}$$

$$\sum_{i=1}^{n}\left(x_{ij}+y_{ij}\right)\leq \phi\cdot z_j \text{ for } j=1,\ldots,m \tag{15}$$

$$x_{ij},y_{ij},z_i\in\{0,1\} \text{ for all } i,j \tag{16}$$

This research proposes a new model based on Model 3, named Model 4 hereafter. The proposed model shares the same Equations (10, 11, 12 and 15), but it proposes a different method for tackling the precedence constraints and uses Equation (17) to replace Equations (13-14).

$$\sum_{j=1}^{m} j\cdot\left(x_{pj}-x_{qj}\right)+\sum_{j=1}^{m}\left(2\cdot m-j\right)\cdot\left(y_{pj}-y_{qj}\right)\leq 0\ \forall(p,q)\in\wp \tag{17}$$

The logic lying behind this new model is illustrated in Fig.2, where each station is divided into two sub-stations. The sub-stations on the entrance side is encoded with 1, 2, …, m and the sub-stations on the exit side is encoded with 2m, 2m-1, …, m+1. The Equation (17) guarantees that a task can be allocated to the current sub-station when all its predecessors have been allocated to the same or former sub-stations. It is obvious that the precedence constraint can be satisfied, and this new model somewhat transfers the UALBP into SALBP with a new cycle time constraint.

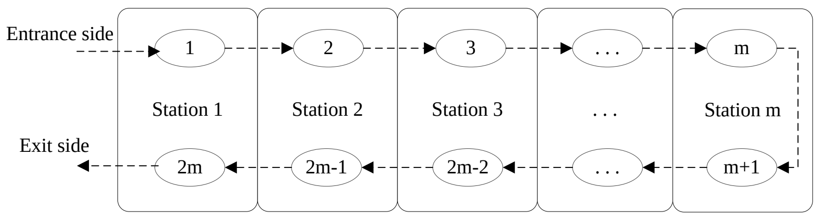


Fig.2 New encode of the stations

Notice that Model 4 refers to the new MILP model in this paper and it is comprised of Equations (10-12), Equation (17) and Equations (15-16). The four models will be further analyzed in Section 2.2.

2.2 Model analysis

All the models address the cycle time constraint properly, whereas Model 1 might achieve solutions conflicted with precedence constraint. This section proves the correctness of the Model 2, Model 3 and the proposed Model 4, and provides the situation where Model 1 will achieve infeasible solutions by enumerating all the possible allocations between two tasks. Suppose that task *p* is the predecessor of task *q*, the possible allocations of the two tasks are listed as follows (including those conflicting with the precedence constraint):

1) Task *p* and task *q* are allocated to the same station, referred to as Case 1.
2) Task *p* and task *q* are both allocated to the entrance side, and there are two situations: task *p* is allocated before task *q*, referred to as Case 2A, and task *q* is allocated before task *p*, referred to as Case 2B.
3) Task *p* and task *q* are both allocated to the exit side, and there are two situations: task *p* is allocated before task *q*, referred to as Case 3A; and task *q* is allocated before task *p*, referred to as Case 3B.
4) Task *p* and task *q* are allocated to the entrance and exit sides. There are two situations: task *p* is allocated to the entrance side and task *q* is allocated to the exit side, referred to as Case 4A, and task *q* is allocated to the entrance side and task *p* is allocated to the exit side, referred to as Case 4B.

Among these occasions, Case 1, Case 2A, Case 3B and Case 4A satisfy the precedence constraint, whereas Case 2B, Case 3A and Case 4B violate the precedence constraint. The correct model must accept the occasions where precedence constraint is satisfied, and deny the occasions where precedence constraint is violated. All the four models accept Case 1, Case 2A, Case 3B and Case 4A, and deny the Case 2B and Case 3A. Nevertheless, constraints (4-5) in Model 1 are reduced to Equations (18-19) when Case 4B occurs and it is obvious that Model 1 cannot deny Case 4B.

$$\sum_{j=1}^{m} j\cdot\left(a_{pj}-a_{qj}\right) \le \psi \; \forall (p,q)\in\wp \qquad (18)$$

$$\sum_{j=1}^{m} j\cdot\left(a_{qj}-a_{pj}\right) \le \psi \; \forall (p,q)\in\wp \qquad (19)$$

The other three models, on the contrary, are able to deny Case 4B. For simplicity, this section mainly focuses on Model 4. Equation (17) is reduced to Equation (20), and finally, Equation (21). Obviously, Equation (21) cannot happen and thus the Case 4B is denied by the proposed Model 4.

$$\sum_{j=1}^{m} j\cdot\left(0-x_{qj}\right)+\sum_{j=1}^{m}\left(2m-j\right)\cdot\left(y_{pj}-0\right)\leq 0 \quad \forall(p,q)\in\wp \tag{20}$$

$$2m-\sum_{j=1}^{m} j\cdot x_{qj}-\sum_{j=1}^{m} j\cdot y_{pj}\leq 0 \quad \forall(p,q)\in\wp \tag{21}$$

In brief, Model 2, Model 3 and the proposed Model 4 are capable of satisfying the precedence constraint, whereas Model 1 might generate infeasible solutions since it cannot deny Case 4B. More performance analysis is provided in Section 5 to compare the speed of these models. Recall that Model 1 is not reflecting some operational constraints in the other models, hence it might produce infeasible solutions. However, the resulting solution can be used as a good lower bound for the actual problem as a relaxation. It is also noticed that both Model 2 and Model 3 have four variables and five constraints. The proposed model, on the contrary, has four variables and four constraints. That is one fewer than both Model 2 and Model 3 as they utilize two expressions to describe the precedence relationship constraint.

3. Proposed methodology

ACO is a powerful technique that has widely been used for solving complex engineering design problems. It mimics the behavior of ants in searching for the shortest paths from their nest to food sources (Sabuncuoglu et al., 2009). This optimization methodology has been applied to various kinds of assembly line balancing problems recently, see for example Boysen and Fliedner (2008), Sabuncuoglu et al. (2009), Baykasoğlu and Dereli (2009), and Zheng et al. (2013). Dorigo and Blum (2005) presented a survey on the ACO algorithms, and specifically, it was shown in Sabuncuoglu et al. (2009) that ACO outperforms simulated annealing algorithm and achieves competing results to ULINO. Hence, this algorithm is selected in this paper.

Different from some other metaheuristic methods, ACO constructively builds feasible solutions by selecting a set of tasks based on pheromone trails and roulette wheel selection to be allocated to the workstations. However, the first task is selected at random rather than utilizing pheromone trails. In addition, the generated task assignment for the current workstation might contain unsatisfying workload and so the idle time of this workstation might be quite large. As known, if less workload is allocated to the former workstations, more workstations to be needed to perform remaining workload, which will eventually lead to a larger number of workstations. Hence, this research presents a variant of the traditional ACO, where two possible improvements are introduced. First, new pheromone trails ($\tau 0_p$) are applied to determine the selection probability of the first task. Second, a new station-oriented decoding procedure is developed where the best one among a set of groups of tasks is selected and assigned to the current station. This station-oriented procedure attempts to eliminate the poor task assignment (having less workload) and increase the possibility of achieving solutions with fewer number of workstations. This procedure and the main segments of the proposed ACO are introduced in the following subsections.

3.1 Modified ACO procedure

The flowchart of the proposed ACO procedure is illustrated in Fig.3. It starts with initializing two sets of pheromone trails, and subsequently, individuals are obtained using the station-oriented procedure, given in Section 3.2. The pheromone trails are updated based on the obtained individuals. In contrary to the traditional ACO, this research employs two kinds of pheromone trails: pheromone trails between tasks, $\tau_{(p,q)}$ (Sabuncuoglu et al., 2009), and new pheromone trails between a fictitious start point and a task, $\tau 0_p$. The new pheromone trails take effect when selecting the first task and it avoids the possible drawback in the original ACO where the first task is randomly selected.

```
Algorithm SACO for UALBP
%s refers to index of an individual and Popsize is the population size
Begin
  Initialize pheromone trails, $\tau 0_p$ and $\tau_{(p,q)}$
  Repeat
    For s=1,…,Popsize
      Obtain individuals using station-oriented procedure
      Achieve the fitness of individual p
    End for
    Update the pheromone trails, $\tau 0_p$ and $\tau_{(p,q)}$
  Until (termination criterion is met)
  Output the best individual
End
```

Fig.3 Flowchart of the modified ACO

The $\tau 0_p$ and $\tau_{(p,q)}$ are updated with Equations (22-25), where Fit_s is the fitness of the individual s, *Popsize* is the population size and ρ is the evaporation rate.

$$\tau 0_p = (1\text{-}\rho)\cdot\tau 0_p + \sum_{s=1}^{Popsize} \Delta\tau 0_p^s \qquad (22)$$

$$\Delta\tau 0_p^s = \begin{cases} 1/Fit_s, & \text{if task } p \text{ is the first operated task in the solution built by ant } s \\ 0, & \text{otherwise} \end{cases} \qquad (23)$$

$$\tau_{(p,q)} = (1\text{-}\rho)\cdot\tau_{(p,q)} + \sum_{s=1}^{Popsize} \Delta\tau_{(p,q)}^s \qquad (24)$$

$$\Delta\tau_{(p,q)}^s = \begin{cases} 1/Fit_s, & \text{if task } q \text{ is operated immediately after task } p \text{ in the solution built by ant } s \\ 0, & \text{otherwise} \end{cases} \qquad (25)$$

The station-oriented procedure is expressed in following sub-section to detail the application of these pheromone trails.

3.2 Station-oriented solution representation

This section illustrates the solution representation in Fig.4, where a set of task assignments are generated for the current station. For achieving each task assignment, the assignable task set is obtained at first. A task is assignable if the finishing time of this task is equal to or the same to the cycle time and all its predecessors or successors have been allocated. Subsequently, this solution generation process introduces two rules to select the assignable tasks. The first task assignment rule gives priorities to the tasks whose finishing time is larger than or equal to the average cycle time (AT) calculated with AT=$\sum_i^n t_i/LB$, where LB is the lower bound calculated with LB=$\lceil\sum_i^n t_i/CT\rceil^+$. The expression $\lceil X\rceil^+$ here denotes the least integer greater than or equal to *X*. The second task assignment rule gives priorities to the tasks whose finishing time is equal to the cycle time. The two task assignment rules achieve the same goal of assigning as much workload as possible to the current station.

```
Algorithm Station-oriented procedure
%l refers to index of a task assignment to current station
%LN is the number of generated task assignments
Repeat
  Open a new station
  For $l=1,\dots,LN$
    Repeat
      Achieve the assignable task set
      If (An assignable task's finishing time is larger than AT)
        Delete the tasks whose finishing times are smaller than AT
        If (An assignable task's finishing time is equal to CT)
          Delete the tasks whose finishing times are smaller than CT
          Obtain the selection probabilities of the remained assignable tasks
          Select a task using the roulette wheel selection scheme
          Update remained capacity of the current station
    Until (no assignable task exists)
  End for
  Select the best task assignment among LN task assignments
Until (all tasks have been allocated)
```

Fig.4 Station-oriented solution representation

The selection of a task among the assignable tasks is another important issue. If no task has been allocated and all the predecessors of the assignable task q have been allocated, the selection probability of task q is set to $\tau 0_q^{\alpha} \cdot w_q^{\beta} \cdot t_q^{\gamma}$, where α, β and γ are three input parameters, and w_q is ranked positional weight of task q (which is the sum of the operation times of task q and all its successors). If at least one task has been allocated, the selected probability of task q is set to $\tau_{(p,q)}^{\alpha} \cdot w_q^{\beta} \cdot t_q^{\gamma}$, where p is the last allocated task. Since a task is also assignable if all its successors have been allocated, ow_q is utilized when all the successors of the assignable task q have been allocated. ow_q is the ranked positional weight of task q in the reverse direction, and it is equal to the sum of the operation times of task q and all its predecessors. On the basis of the selection probabilities of the remained assignable tasks, a task is selected using the roulette wheel selection scheme. Subsequently, the selected task is allocated to the current station, and this procedure is repeated until no assignable task exists.

After achieving a set of LN task assignments, the next job is selecting the best one as the current task assignment. This research selects the task assignment with a large value of $\sum_{i\in FS}\left(t_i + a\cdot w_i/CT + b\cdot F_i - c\right) + \sum_{i\in RS}\left(t_i + a\cdot ow_i/CT + b\cdot oF_i - c\right)$, where FS (RS) is the set of tasks allocated to the current station in the forward (reverse) direction or their predecessors (successors) have been allocated before allocating them. F_i (oF_i) refers to the number of immediate predecessors (successors) of task i. Parameter a, b and c are three input parameters, and the largest workload is selected when the values of three parameters are all set to 0.0. The rationale of this formula and the effect of each part are clarified as follows. The term t_i preserves the larger workload, the term $a \cdot w_i / CT$ selects the tasks whose successors have a large total operation time, the term $b \cdot F_i$ favors the tasks which will make more tasks assignable, and the term $-c$ selects the tasks with larger operation times. The values of a, b and c are set to be much smaller than 1.0 in this paper so that the latter parts take effect only when the allocated workloads are equal.

In brief, this station-oriented procedure generates a set of possible task assignments (or groups of tasks), and then selects the best one as the current task assignment. The basic logic is similar to the maximum workload rule in Scholl and Klein (1999) and the graded objectives in Li et al. (2017), where both allocate more workloads to the former stations.

3.3 Illustrated example

This section illustrates the station-oriented solution representation by solving a large-sized test problem, namely Tonge-70, where detailed operation times and precedence relationships are exhibited in Table 1.

Table 1 Operation times and precedence relationships of tested problem

Tasks	Operation time	Successors	Tasks	Operation time	Successors
1	17	2, 41, 69, 70	36	40	37
2	66	3	37	2	38
3	54	4, 68	38	1	39
4	52	6, 7	39	3	40
5	6	6, 24, 30	40	13	42
6	88	8	41	16	42
7	21	8	42	25	43
8	128	12	43	21	50
9	68	10	44	43	45
10	70	11	45	30	46
11	85	12	46	83	47
12	21	13, 14	47	89	50
13	134	23	48	56	49
14	135	23	49	59	50
15	94	16	50	43	-
16	90	17, 18	51	11	52
17	50	19	52	26	54
18	143	19	53	44	54
19	19	20, 22, 57	54	121	55
20	54	21	55	38	-
21	50	23	56	68	-
22	40	23	57	22	58
23	73	25, 31, 33	58	7	59
24	12	25	59	16	60
25	152	26, 27, 28, 29	60	32	-
26	42	35	61	25	65
27	45	35	62	27	63
28	74	35	63	156	64
29	26	35	64	28	65, 66, 67
30	11	31	65	15	-
31	31	32	66	26	-
32	50	35	67	18	-
33	102	34	68	72	-
34	46	35	69	23	-
35	35	36, 44, 48, 51, 53, 56, 60, 61, 62	70	27	-

The illustrated case has 70 tasks and the cycle time is fixed to 527 units. Supposed that the number of task assignment or LN is set to 10, detailed station-oriented representation is illustrated in Table 2. In this table, workload means the allocated workload or the total operation time of allocated tasks. Value means the value of $\sum_{i\in FS}\left(t_i+a\cdot w_i/CT+b\cdot F_i\text{-}c\right)+\sum_{i\in RS}\left(t_i+a\cdot ow_i/CT+b\cdot oF_i\text{-}c\right)$, where a, b and c are set to 0.0, 0.0 and 0.3 respectively.

Table 2 Illustrated station-oriented representation

Station	Task assignment	Task assignment	Workload	Values
1	1	65,56,60,50,9,67,70,47,10,46,5	519	518.67
	2	15,67,9,5,60,10,66,70,16,11,30	527	526.67
	3	9,65,68,66,70,56,15,1,16,17	527	526.7
	…	…	…	…
	Selected	9,65,68,66,70,56,15,1,16,17	**527**	**526.7**
2	1	61,5,55,54,10,30,18,67,11	517	516.73
	2	50,61,47,18,2,49,46,19	527	526.76
	3	5,60,61,2,3,4,50,69,10,18,30	525	524.67
	…	…	…	…
	Selected	50,61,47,18,2,49,46,19	**527**	**526.76**
3	1	60,3,22,48,20,21,45,55,57,44,43,10,59	526	525.61
	2	43,67,42,48,60,5,55,59,54,52,45,64,40,24,10,58,39, 38,37	525	524.43
	3	57,48,10,67,43,60,69,3,4,20,22,11	527	526.64
	…	…	…	…
	Selected	57,48,10,67,43,60,69,3,4,20,22,11	**527**	**526.64**
4	1	42,59,5,55,6,54,45,41,64,63	524	523.7
	2	45,64,5,58,63,41,62,21,55,6,44,42,40	527	526.61
	3	5,45,58,6,7,64,42,21,44,30,55,63,24	515	514.61
	…	…	…	…
	Selected	45,64,5,58,63,41,62,21,55,6,44,42,40	**527**	**526.61**
5	1	30,7,8,39,24,59,54,12,14,53,38,37	515	514.64
	2	7,39,8,54,59,24,52,53,12,14	527	526.7
	3	54,7,53,59,52,8,30,51,12,24,39,38,37,36,35,29	518	517.52
	…	…	…	…
	Selected	7,39,8,54,59,24,52,53,12,14	**527**	**526.7**
6	1	51,13,23,38,30,25,26,33	526	525.76
	2	30,13,38,51,23,31,33,25,37	517	516.73
	3	51,30,13,23,31,38,25,33,37	517	516.73
	…	…	…	…
	Selected	51,13,23,38,30,25,26,33	**526**	**525.76**
7	1	34,31,32,28,29,37,36,35,27	349	348.73
	2	29,37,34,31,36,28,27,32,35	349	348.73
	3	29,31,32,37,28,27,36,34,35	349	348.73
	…	…	…	…
	Selected	34,31,32,28,29,37,36,35,27	**349**	**348.73**

In the station-oriented representation, a total of 10 task assignments are achieved at first for station 1, and then the corresponding values are calculated. The task assignment with the largest value is selected as the current task assignment, namely task 9, 65, 68, 66, 70, 56, 15, 1, 16 and 17. After determining the task assignment for station 1, the same procedure is executed for station 2. Again the task assignment with the largest value is selected as the current task assignment, namely task 50, 61, 47, 18, 2, 49, 46, and 19. This procedure is repeated for the remaining workstations and terminated when all the tasks have been allocated. It is also observed that this station-oriented representation allocates 527, 527, 527, 527, 527, 526 and 349 units workload to stations 1, 2, 3, 4, 5, 6 and 7, respectively. It is clear that more workload is allocated to former stations, and the less remained workload is endured by latter stations.

4. Computational study

This section first compares the summarized existing models with the newly developed model (given in Section 2.1), and later presents a comparative study on the proposed algorithm using well-known benchmarks.

4.1 Model comparison

Since Model 1 is incorrect and the comparison between Model 1 and Model 2 has been presented in Fattahi and Turkay (2015), this section mainly focuses on Model 2, Model 3 and the proposed model, namely Model 4. Table 3 presents the results of three models. All the models are coded into the Cplex solver of GAMS and terminated when an optimal solution is achieved or elapsed CPU time reaches 1000 seconds. Notice that this table only presents the results for problems containing 45, 53 and 58 tasks since all the models can solve the smaller-size problems very quickly and cannot achieve the optimal solution or prove the optimally of the achieved solution within the acceptable time when solving the large-size problems. Detailed precedence diagrams of the tested problems are available at http://assembly-line-balancing.mansci.de/wp-content/uploads/2017/01/Scholl-1993-ALBData.pdf. The initial station numbers are set to 12 for Kilbridge and Hahn and 40 for Warnecke. In this table, Nt is the number of tasks, $|\wp|$ is the number of arcs in the precedence diagram and d is the network density calculated with $d = |\wp|/Nt$ (Fattahi & Turkay, 2015). CT refers to the cycle time and iteration means the iteration time before the search process terminates.

Table 3 Computational results by three models

Problem	Nt	$\|\wp\|$	d	CT	Model	Single equations	Single variables	Result	Iteration	Time (s)
Kilbridge	45	62	1.38	56	Model 2	192	598	10	72920	3.69
					Model 3	192	1093	10	9897	0.76
					Model 4	131	1093	10	4286	**0.71**
				57	Model 2	192	598	10	2497	0.6
					Model 3	192	1093	10	13668	0.93
					Model 4	131	1093	10	2299	**0.59**
				62	Model 2	192	598	9	24599	1.23
					Model 3	192	1093	9	20250	1.14
					Model 4	131	1093	9	6833	**0.71**
Hahn	53	82	1.55	2004	Model 2	242	702	No solution		1000
					Model 3	242	1285	**8**	13328488	1000
					Model 4	160	1285	**8**	18409505	1000
				2338	Model 2	242	702	7	16645500	1000
					Model 3	242	1285	7	14030156	1000
					Model 4	160	1285	7	19291955	1000
				2806	Model 2	242	702	6	17070195	1000
					Model 3	242	1285	6	12190233	1000
					Model 4	160	1285	6	19409416	1000
				3507	Model 2	242	702	5	17531451	1000
					Model 3	242	1285	5	11994373	1000
					Model 4	160	1285	5	17603194	1000
				4676	Model 2	242	702	3	2297999	**143.13**
					Model 3	242	1285	3	2615731	224.67
					Model 4	160	1285	3	11398621	611.43
Warnecke	58	70	1.21	54	Model 2	279	2419	32	8945461	1000
					Model 3	279	4681	**31**	7307103	1000
					Model 4	209	4681	**31**	8462699	1000
				56	Model 2	279	2419	No solution		1000
					Model 3	279	4681	**29**	117583	1000
					Model 4	209	4681	30	9946370	1000
				58	Model 2	279	2419	29	11247631	1000

	Model 3	279	4681	29	7603313	1000
	Model 4	209	4681	**28**	1441324	155.32

*Best results or the smallest CPU time in bold.

From Table 3, it is observed that the numbers of single equations are 192, 192 and 131 for Model 2, Model 3 and Model 4, respectively, regarding Kilbridge-45 with the cycle time of 56 units. The corresponding numbers of single variables are 598, 1093 and 1093, respectively. Clearly, Model 2 and Model 3 have the same number of single equations for this case, whereas Model 4 has a smaller number of single equations. In addition, Model 3 and Model 4 share the same number of single variables, whereas Model 2 has fewer single variables. This situation suits all the other cases solved. Another finding is that these three models obtain the same station numbers for most of the tested cases. Nevertheless, Model 2 cannot achieve the feasible solutions for two cases within 1000 seconds.

Regarding the utilized CPU time, all the models can solve the problem with 45 tasks within 10 seconds, whereas they cannot achieve the optimal solution or prove the optimality of the achieved solution within 1000 seconds for most cases. As for the iteration speed, it is observed that Model 4 iterates for the largest times within 1000 seconds for most cases, and Model 3 iterates for the smallest time. Consequently, the proposed Model 4 has fewer equations than Model 2 and Model 3, and has the same variables as Model 3. Model 4 iterates faster than Model 3 for all the cases and faster than Model 2 for most cases. Although Model 4 iterates fast, it achieves similar results with Model 2 and Model 3 and the superiority of Model 4 is not clear. But it is still sufficient to state that Model 4 is an alternative model for U-type assembly line balancing, as it iterates fast and achieve competitive results.

4.2 Comparative study

This section presents a comparative study where the performance of the proposed SACO is compared to those of the five published metaheuristics and one exact method, ULINO (Scholl & Klein, 1999). The metaheuristics are simulated annealing (SA) method (Erel et al., 2001), three modified ACO methods (Sabuncuoglu et al., 2009) and another ACO method (Baykasoğlu & Dereli, 2009).

To differentiate ACO methods, the three methods by Sabuncuoglu et al. (2009) are marked with ACSm-Sabuncuoglu, ACO−Version 1-Sabuncuoglu and ACO−Version 2-Sabuncuoglu. The ACO method by Baykasoglu and Dereli (2009) is marked with ACO-Baykasoğlu. The ULINO by Scholl and Klein (1999), to our best knowledge, is still the best performer for UALBP which achieves the maximum number of optimal solutions.

Apart from the aforementioned four other published methods, four variants of ACO are also included in this research to evaluate the performance of the improvements. These variants are presented as follows. Among the four variants, ACO1 is the original ACO and ACO2 differs from ACO1 in that new pheromone trails $\tau 0_p$ are employed. ACO3 differs from ACO2 in utilizing task assignment rule, which is proposed to test the performance of the developed task assignment rule. ACO4 inherits all features of the proposed SACO but the values of parameter *a*, *b* and *c* are set to 0.0. Specifically, ACO4 selects the task assignment with the largest allocated workload.

ACO1: The new pheromone trails $\tau 0_p$ are not employed and no task assignment rule or station-oriented procedure are applied.

ACO2: The new pheromone trails $\tau 0_p$ are employed whereas no task assignment rule or station-oriented procedure are applied.

ACO3: The new pheromone trails $\tau 0_p$ and task assignment rule are employed whereas the station-oriented procedure is not applied.

ACO4: All the new pheromone trails $\tau 0_p$, task assignment rule and station-oriented procedure are employed, but the values of parameter a, b and c are set to 0.0.

There are two benchmark sets available to test the performance of these methods: the set summarized in Scholl and Klein (1999) and the new one generated in Otto et al. (2013). The benchmark set summarized in Scholl and Klein (1999) has been widely applied whereas no paper has adopted the benchmark set by Otto et al. (2013) for the considered problem. In order to compare the published methods, this research mainly presents the computational results on the benchmark set in Scholl and Klein (1999). Regarding this benchmark set, there are a total number of 269 cases and all of them are solved in this study. Detailed precedence diagrams of all the tested problems are available at http://assembly-line-balancing.mansci.de/wp-content/uploads/2017/01/Scholl-1993-ALBData.pdf. Determining a proper termination criterion is an important issue. As in Scholl and Klein (1999), the methods terminate when the elapsed CPU time reaches 500 seconds per instance.

For metaheuristics, parameter values play an important role in their final performances. This paper proposes the Taguchi method (Mozdgir et al., 2013) for parameter calibration. For simplicity, this section only presents the parameter calibration process of ACO1. There are six parameters to be determined: population size, initial pheromone trails, evaporation coefficient ρ, and weighting parameters α, β and γ. Each parameter is tested at three levels, so there are 27 combinations of the parameter values in the orthogonal table. Detailed orthogonal table and the levels of parameters are presented in Table A1 (see Appendix). Twenty cases of the largest problem Scholl-297 with 297 tasks are employed for parameter calibration, and the achieved station number is transferred into relative percentage deviation or RPD. Relative percentage deviation is calculated with RPD=$100\cdot\left(Fit_{some}\text{-LB}\right)/\text{LB}$, where LB is the lower bound on station number reported in Scholl and Klein (1999) and Fit_{some} is the station number found by an algorithm. After carrying out all the experiments, the average RPD value of twenty cases are selected as the response variable and the multi-factor analysis of variance (ANOVA) test is executed following Mozdgir et al. (2013), Tang et al. (2016) and Li et al. (2017), among others. Detailed ANOVA results are omitted in the section for space reason, but the average RPD plot and the corresponding S/N ratio plot are presented in Fig. A1 and Fig. A2 (see Appendix). Other tested methods are calibrated in the same methods, and the tested parameter values are presented in Table A2 (see Appendix). The detailed applications of the Taguchi method and ANOVA analysis are available upon request.

The achieved best results and the average results achieved within twenty-times run are reported in Table 4. Notice that ULINO belongs to the exact methods and it can obtain the same results in repeated runs. ACO applies a stochastic search mechanism due to the randomness involved in its nature. To maintain robustness, each test problem was run twenty-times and the results are reported in Table 4 together with some performance metrics achieved in doing so. In Table 4, '# instances' is the number of tested instances, '# Opt' is the number of instances that optimal solutions are achieved at least once within twenty-times run. As each test problem was run twenty-times, this research utilizes RPD_{Best} to denote the best RPD among the twenty RPD values within twenty-times run for each instance and RPD_{Avg} to denote the average RPD within twenty-times run for each instance. RPD_{Best}-*Avg* and RPD_{Best}-*Max* are the average and maximum of the RPD_{Best} values belonging to all test cases, respectively. RPD_{Avg}-*Avg* is the average of the RPD_{Avg} values of all the tested cases, and RPD_{Avg}-*Max* is the maximum of the RPD_{Avg} values of all the tested cases.

Table 4 Comparison between ULINO and applied ACO methods

Method	Instance	Talbot	Hoffmann	Scholl	Combined
	# instances	64	50	168	269
ULINO	*# Opt*	60	32	150	233
	RPD_{Best}-*Avg*	0.39	1.99	0.08	0.59
	RPD_{Best}-*Max*	7.14	10	5.26	10
ACO1	*# Opt*	58	26	73	148
	RPD_{Best}-*Avg*	0.73	2.68	2.13	1.91
	RPD_{Best}-*Max*	14.29	7.69	8.33	14.29
	RPD_{Avg}-*Avg*	0.74	2.85	2.37	2.09
	RPD_{Avg}-*Max*	14.29	7.92	10.00	14.29
ACO2	*# Opt*	58	25	78	152
	RPD_{Best}-*Avg*	0.73	2.84	1.97	1.84
	RPD_{Best}-*Max*	14.29	8.33	8.33	14.29
	RPD_{Avg}-*Avg*	0.75	2.88	2.34	2.08
	RPD_{Avg}-*Max*	14.29	8.33	8.89	14.29
ACO3	*# Opt*	59	28	116	194
	RPD_{Best}-*Avg*	0.61	2.41	1.11	1.19
	RPD_{Best}-*Max*	14.29	7.69	6.25	14.29
	RPD_{Avg}-*Avg*	0.62	2.66	1.32	1.38
	RPD_{Avg}-*Max*	14.29	7.69	7.92	14.29
ACO4	*# Opt*	63	39	161	251
	RPD_{Best}-*Avg*	0.09	1.07	0.15	0.29
	RPD_{Best}-*Max*	5.88	6.25	4.76	6.25
	RPD_{Avg}-*Avg*	0.30	1.47	0.16	0.37
	RPD_{Avg}-*Max*	6.79	6.79	4.76	6.79
SACO	*# Opt*	**64**	**43**	**161**	**255**
	RPD_{Best}-*Avg*	**0.00**	**0.61**	**0.15**	**0.21**
	RPD_{Best}-*Max*	**0.00**	**5.00**	**4.76**	**5.00**
	RPD_{Avg}-*Avg*	**0.06**	**0.85**	**0.15**	**0.25**
	RPD_{Avg}-*Max*	**3.53**	**5.00**	**4.76**	**5.00**

*Best results in bold

It has been observed during the computational study that the ACO variants whose performances have been compared to that of SACO cannot achieve the same results with SACO even when the CPU time increases. Therefore, the results have been compared in terms of both the best results and the relative percentage deviation metrics. As for assembly line balancing problem, the number of the achieved optimal solutions is an important indicator to evaluate the performance of the algorithms, and thus this research mainly compares the number of achieved optimum solutions following the common tendency in the literature (e.g. see Sabuncuoglu et al. (2009)). It is observed from Table 4 that the proposed SACO is capable of finding optimal solutions for 255 cases, ranking first among the compared methods. Specifically, SACO outperforms ACO4, ULINO and ACO3 for 4, 22 and 61 cases out of 269 cases, respectively. ACO4 is the second-best performer which achieves the optimal solutions for 251 cases.

ULINO achieves the third largest number of found optimal solutions, and ACO3 achieves the fourth largest number. ACO2 and ACO1 are the two worst performers and ACO2 shows a small superiority over ACO1. This finding suggests that the utilization of the new pheromone trails is reasonable. In addition, ACO3 shows a clear advantage over the ACO1 and ACO2, which indicates the superiority of utilizing task assignment rules. The proposed SACO is also the best performer when utilizing the RPD_{Best}-*Avg*, RPD_{Best}-*Max*, RPD_{Avg}-*Avg* and RPD_{Avg}-*Max* as the evaluation metrics. Again, ACO4 and ULINO are the second and third best performers when utilizing these evaluation metrics. The superiority of the SACO over the ULINO on the number of achieved optimal solutions suggests that the proposed SACO is highly effective for UALBP, and can be regarded as the state-of-the-art metaheuristic for UALBP. In fact, the high performance of the proposed SACO is mainly attributed to the utilization of the station-oriented procedure. This station-oriented procedure selects the best one among a set of task assignments (or groups of tasks), which ensures that the former workstations endures more workload and increase the possibility of achieving solutions with fewer workstations. In addition, this procedure also allocates the tasks with larger operation time at first and makes more tasks assignable for the latter workstations by allocating the tasks with more successors in the forward direction (predecessors in the reverse direction) preferably.

A more thorough comparative study is presented in Table 5, which exhibits the achieved best results by proposed ACO methods and the best results published by five metaheuristics and an exact method, ULINO. Notice that the published five metaheuristics only tackle a portion of the benchmark problems, and thus this table summarizes the number of solved instances and the number of instances solved optimally. This table only calculates the percentage of the instance solved optimally in the last column. If the algorithms are ranked in decreasing order of the percentage values, the proposed SACO ranks first and ACO4 and ULINO rank the second the third. The ACO-Version 2-Sabuncuoglu ranks the fourth which is the best performer among the ACO methods which do not utilize station-oriented decoding. Among all these methods, ACO-Baykasoğlu is the worst performer which only achieves the optimal solutions for 90 cases out of 232 tested cases. In addition, if the number of instances not solved optimally is considered, there are only 14 and 18 cases which are not solved optimally by SACO and ACO4. Nevertheless, there are 60 and 142 cases not solved optimally by SA and ACO-Baykasoğlu even though they solve fewer instances. All these computational results show that the proposed SACO outperforms the compared ones regarding the number of achieved optimal solutions.

Table 5 Comparison between published methods and applied ACO methods

Methods	Talbot		Hoffmann		Scholl		Combined cases		
	# instances	# Opt	# instances	# Opt	# instances	# Opt	# instances	# Opt	Percent
SA	-	-	-	-	168	112	187	127	67.91
ACO-Baykasoğlu	64	55	-	-	168	35	232	90	38.79
ACSm-Sabuncuoglu	-	-	-	-	-	-	190	81	42.63
ACO-Version 1-Sabuncuoglu	-	-	-	-	-	-	190	108	56.84
ACO-Version 2-Sabuncuoglu	-	-	-	-	-	-	190	144	75.79
ULINO	64	60	50	32	168	150	269	233	86.62
ACO1	64	58	50	26	168	73	269	148	55.02
ACO2	64	58	50	25	168	78	269	152	56.51

ACO3	64	59	50	28	168	116	269	194	72.12
ACO4	64	63	50	39	168	161	269	251	93.31
SACO	64	64	50	43	168	161	269	255	**94.80**

*Best results in bold

Table 6 also shows the results of the hard instances from the tested 269 cases, where Nt is the number of tasks, $|\wp|$ is the number of arcs in the precedence diagram, d is the network density, $d = |\wp|/Nt$, and CT is the cycle time (Fattahi & Turkay, 2015). The results of ACO methods are the best solution achieved in twenty-times independent runs. In this table, SACO updates the results by ULINO for 21 cases. Since the newly achieved station number is equal to the lower bound provided by ULINO, it is proper to state that the optimal solutions of these 21 cases are first achieved. When the results obtained by ACO methods are compared, it is observed that ACO1, ACO2 and ACO3 cannot achieve the optimal solutions for all the cases of Barthol2-148 and Scholl-297 test cases. These findings suggest that the original ACO methods, ACO1, ACO2 and ACO3, are ineffective when solving large-size problems.

Table 6 Comparison on challenging instances

Graph	Nt	$\|\wp\|$	*d*	CT	ULINO	ACO1	ACO2	ACO3	ACO4	SACO
Warnecke	58	70	1.21	54	[30,31]	31	31	31	31	31
				62	[26,27]	27	27	27	26	**26**
				65	[24,25]	25	25	25	25	25
				68	[23,24]	24	24	24	23	**23**
				71	[22,23]	23	23	23	23	23
				74	[21,22]	22	22	22	22	22
				82	[19,20]	20	20	20	19	**19**
Tonge	70	86	1.23	160	[22,23]	23	23	23	22	**22**
				176	[20,21]	21	21	21	20	**20**
Wee-mag	75	87	1.16	47	[32,33]	33	33	32	32	**32**
				49	[31,32]	32	32	32	32	32
				50	[31,32]	32	32	32	32	32
Arcus 1	83	113	1.36	3786	[21,22]	21	21	21	21	**21**
Mukherje	94	181	1.93	176	[24,25]	25	25	25	24	**24**
Arcus 2	111	176	1.59	5785	[26,27]	27	27	27	27	27
				6016	[25,26]	26	26	26	26	26
				6267	[24,25]	25	25	25	25	25
				6540	[23,24]	24	24	24	24	24
				6837	[22,23]	23	23	23	23	23
				7162	[21,22]	22	22	22	22	22
				7520	[20,21]	21	21	21	21	21
				7916	[19,20]	20	20	20	20	**19**
				8356	[18,19]	19	19	19	19	**18**
				8847	[17,18]	18	18	18	18	**17**
				9400	[16,17]	17	17	17	17	**16**
				10027	[15,16]	16	16	16	15	**15**
				10743	[14,15]	15	15	15	14	**14**
				11570	[13,14]	14	14	14	13	**13**
Barthol2	148	175	1.18	85	[50,51]	51	51	51	50	**50**
				89	[48,49]	49	49	48	48	**48**
				93	[46,47]	47	47	46	46	**46**
				97	[44,45]	45	45	44	44	**44**
Scholl	297	423	1.42	1394	[50,51]	51	51	51	50	**50**
				1422	[49,50]	50	50	50	50	50
				1452	48	49	49	49	48	48
				1483	47	48	48	48	47	47
				1515	[46,47]	47	47	47	46	**46**
				1548	45	46	46	46	45	45
				1584	44	45	45	45	44	44
				1620	43	44	44	44	43	43
				1659	42	43	43	43	42	42

1699	41	42	42	42	41	41
1742	40	41	41	41	40	40
1787	39	40	40	40	39	39
1834	38	39	39	39	38	38
1883	37	38	38	38	37	37
1935	36	37	37	37	36	36
1991	35	36	36	36	35	35
2049	34	35	35	35	34	34
2111	33	34	34	34	33	33
2177	32	33	33	33	32	32
2247	31	32	32	32	31	31
2322	30	31	31	31	30	30
2402	29	30	30	30	29	29
2488	28	29	29	29	28	28
2580	27	28	28	28	27	27
2680	26	27	27	27	26	26
2787	25	26	26	26	25	25

*New found upper bounds or optimal solutions in bold.

To have a better observation of the evolution process of the tested algorithms, Fig.5 illustrates the average workstation numbers achieved by the tested algorithm during evolution when solving Tonge-70 with a cycle time fixed to 168 units. From this figure, it is observed that none of the tested ACO methods shows clear convergence with the increasing CPU time. In fact, this interesting situation is attributed to the selection probabilities of tasks and the characteristic of the considered problem. As presented in Section 3.2, a task is selected based on the selection probabilities of tasks using the roulette wheel selection scheme, where ranked positional weight and operation time are also included in the selection strategy as heuristic information. While the heuristic information increases the exploration capacity and helps achieve diverse individuals, the proposed methods cannot show clear convergence. Additionally, the workstation number, which is determined by the task assignments on all workstations, is optimized. The optimal solution of the tested case is achieved only when all the workstations endure enough workload. However, the randomness in the task selection makes the probability of achieving optimal solution very low, and hence the population cannot convergence to the optimal or near optimal solution.

Still, it is clear that SACO achieves the smallest values of the average workstation numbers. ACO1 and ACO2 obtain the largest values of the average workstation numbers. This demonstrates that the proposed ACO methods are capable of achieving diverse solutions and SACO outperforms the compared methods during the evolution process regarding the average workstation numbers.

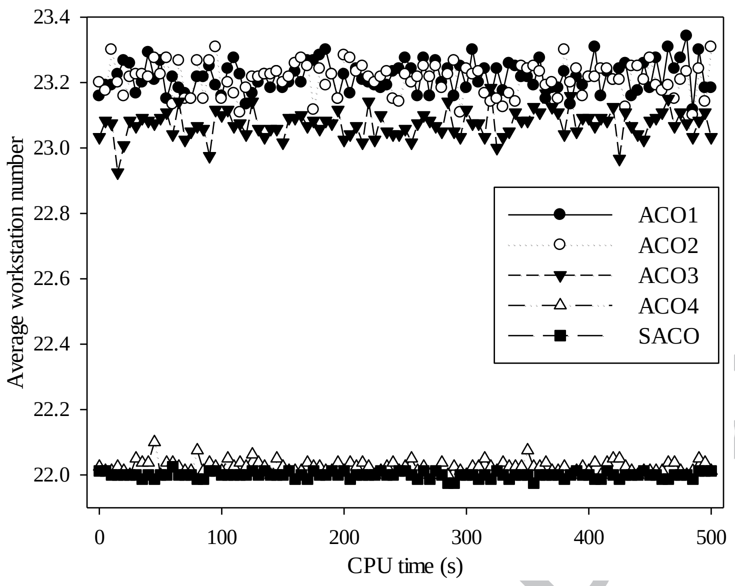


Fig.5 Average workstation number during the evolution process

From the above computational study, it is clear that SACO has a promising solution building capacity regardless of the problem size. As seen in Table 6, SACO maintains its competitiveness even the number of tasks increases from one problem to another. Thus, it is established that the proposed SACO is quite effective for UALBP and capable of achieving solutions with fewer workstations.

The reduction in the number of opened workstations yields a decrease in the number of operators working in the line composed of those workstations. As known, U-type assembly lines are widely utilized in assembly industry to achieve high flexibility and productivity and the number of workers utilized on this line is the main expense. Thus, the reduction of the number opened workstations results in a decrease in labor cost. Therefore, SACO can improve line efficiency and reduce labor cost in U-shaped assembly lines. Line managers can undoubtedly use SACO for balancing U-shaped lines in real-life implementations to reduce their costs.

5. Conclusions and future research

This research addresses the U-type assembly line balancing problem to minimize the number of opened workstations. A new mixed-integer linear programming model is developed which utilizes one expression to describe the precedence constraint. The validity of this model and three compared models are analyzed by enumerating the possible allocations of two tasks. The proposed model and the two out of three compared ones are correct while one out of the three models is wrong. A comparative study on these models demonstrates that the proposed model executes more number of iterations within the same CPU time (1000 seconds) for most cases and it has the capability of achieving competing results.

To tackle the large-size problems, this research develops a station-oriented ant colony optimization. This modified ant colony optimization differs from the original ones in two aspects. New pheromone trails $\tau 0_p$ are applied to determine the selection probability of the first task. A new station-oriented decoding is developed to achieve solutions where the best one among a set of task assignment groups is selected for the current station. The proposed method is compared with four variants of ant colony optimization, exact method ULINO and five published metaheuristics. Computational results demonstrate the proposed method is the best performer among the tested algorithms. Specifically, this newly developed method achieves the optimal solutions of 255 cases out of 269 cases and it outperforms the current best method ULINO for 21 cases regarding the best station number achieved. The proposed method is capable of reducing the number of workstations, which yields to a decrease in labor cost eventually thanks to the reduction in the number of operators. The methodology proposed in this research can easily be adopted by practitioners, i.e. line managers. Thus, a considerable number of workstations can be saved when constructing a new U-shaped line as well as balancing an existing line. However, the model still cannot solve large-size problems optimally within acceptable time and the algorithm needs several parameters to be calibrated. Also, the production environment may contain more sophisticated applications and/or constraints to be considered, such as customized demands requiring mixed-model production and the utilization of robots requiring their optimal assignment to workstations. Future research might consider focusing on these issues. The model can also be extended to balance other U-type assembly lines, including robotic U-type assembly lines, mixed-model U-type assembly lines and U-type assembly lines with model sequencing. Thus, this new method for tackling precedence constraint can help building the models for such problems. Furthermore, the proposed station-oriented method has wider applications due to its high performance observed in this study. This method might be applied to solve miscellaneous two-sided and parallel assembly line balancing problems. Even for type II assembly line balancing problem, this method might produce promising results utilizing an iterative search mechanism. It is also worthy to implement some other recent and effective metaheuristics.

Appendix

Nomenclature

i, p, q	: The task indices
n, Nt	: The total number of tasks
j	: The workstation index
m	: The upper bound on the number of stations
z_j	: A binary variable, $z_j=1$ if station j is utilized; $z_j=0$, otherwise
a_{ij}	: A binary variable, $a_{ij}=1$ if task i is allocated to station j; $a_{ij}=0$, otherwise
t_i	: The operation time of task i
CT	: The cycle time
ψ	: A very large positive number
u_i	: A binary variable, $u_i=0$ if task i is allocated to the entrance side; $u_i=1$ if task i is allocated to the exit side
$\wp=\{(p,q)\}$	: The set of paired tasks, where task p is the immediate predecessor of task q
x_{ij}	: A binary variable, $x_{ij}=1$ if task i is allocated to the entrance side of station j; $x_{ij}=0$ otherwise
y_{ij}	: A binary variable, $y_{ij}=1$ if task i is allocated to the exit side of station j; $y_{ij}=0$ otherwise
$\tau0_p$	: The pheromone trail between a fictitious start point and task p

$\tau_{(p,q)}$	: The pheromone trail between tasks *p* and *q*
Popsize	: Population size
s	: The index of an individual in the population (*s=1,…,Popsize*)
Fit_s	: The fitness of individual *s*
ρ	: Evaporation rate
AT	: Average cycle time
LB	: Lower bound, LB=$\lceil\sum_i^n t_i/CT\rceil^+$ where the expression $\lceil X\rceil^+$ denotes the least integer greater than or equal to *X*.
α, β, γ	: Input parameters
w_q	: The ranked positional weight of task *q* (the sum of the operation times of task *q* and all its successors)
ow_q	: The ranked positional weight of task *q* in the reverse direction (the sum of the operation times of task *q* and all its predecessors)
LN	: The number of generated task assignments
l	: The index of a task assignment to current station (*l=1,…,LN*)
FS	: The set of tasks allocated to the current station in the forward direction
RS	: The set of tasks allocated to the current station in the reverse direction
F_i	: The number of immediate predecessors of task *i*
oF_i	: The number of immediate successors of task *i*
a, b, c	: Input parameters
$\lvert\wp\rvert$	: The number of arcs in the precedence diagram
d	: The network density calculated with $d = \lvert\wp\rvert/Nt$
Fit_{some}	: The station number found by an algorithm
RPD	: Relative percentage deviation, RPD=100$\cdot\left(Fit_{some}\text{-LB}\right)/\text{LB}$
# Opt	: The number of instances that optimal solutions are achieved at least once within a certain number of iterations
RPD_{Best}	: The best *RPD* among those obtained within twenty-times run for each instance
RPD_{Avg}	: The average of the twenty *RPD* values obtained within twenty-times run for each instance
RPD_{Best}-*Avg*	: The average of the RPD_{Best} values of all the instances tested
RPD_{Best}-*Max*	: The maximum of the RPD_{Best} values of all the instances tested
RPD_{Avg}-*Avg*	: The average of the RPD_{Avg} values of all the instances tested
RPD_{Avg}-*Max*	: The maximum of the RPD_{Avg} values of all the instances tested

Table A1 Orthogonal table of parameter levels

Population size	Initial pheromone trails	Evaporation coefficient	α	β	γ	Average RPD
40	1	0.1	0.1	0	0	2.62109
40	1	0.1	0.1	0.1	0.1	2.52109
40	1	0.1	0.1	0.2	0.2	2.52109
40	5	0.2	0.2	0	0	2.72313
40	5	0.2	0.2	0.1	0.1	2.52109
40	5	0.2	0.2	0.2	0.2	2.52109
40	10	0.3	0.3	0	0	2.93368

40	10	0.3	0.3	0.1	0.1	2.62109
40	10	0.3	0.3	0.2	0.2	2.52109
80	1	0.2	0.3	0	0.1	2.62109
80	1	0.2	0.3	0.1	0.2	2.52109
80	1	0.2	0.3	0.2	0	2.52109
80	5	0.3	0.1	0	0.1	2.52109
80	5	0.3	0.1	0.1	0.2	2.52109
80	5	0.3	0.1	0.2	0	2.52109
80	10	0.1	0.2	0	0.1	2.52109
80	10	0.1	0.2	0.1	0.2	2.52109
80	10	0.1	0.2	0.2	0	2.52109
120	1	0.3	0.2	0	0.2	2.52109
120	1	0.3	0.2	0.1	0	2.52109
120	1	0.3	0.2	0.2	0.1	2.52109
120	5	0.1	0.3	0	0.2	2.52109
120	5	0.1	0.3	0.1	0	2.52109
120	5	0.1	0.3	0.2	0.1	2.52109
120	10	0.2	0.1	0	0.2	2.52109
120	10	0.2	0.1	0.1	0	2.52109
120	10	0.2	0.1	0.2	0.1	2.52109

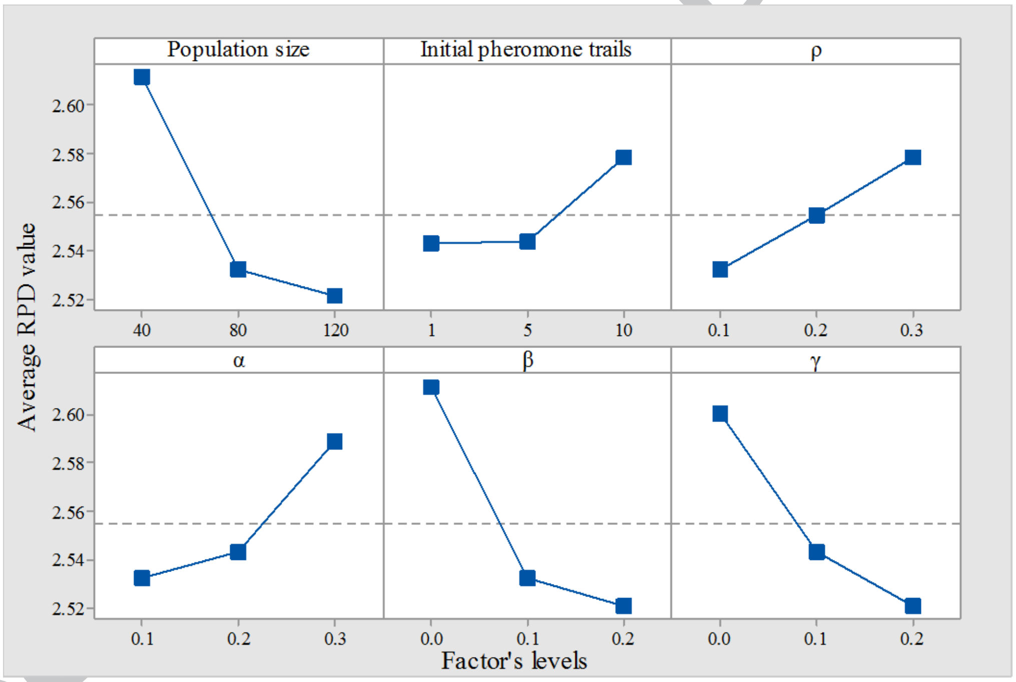


Fig. A1 The average RPD plot for each level of the parameters.

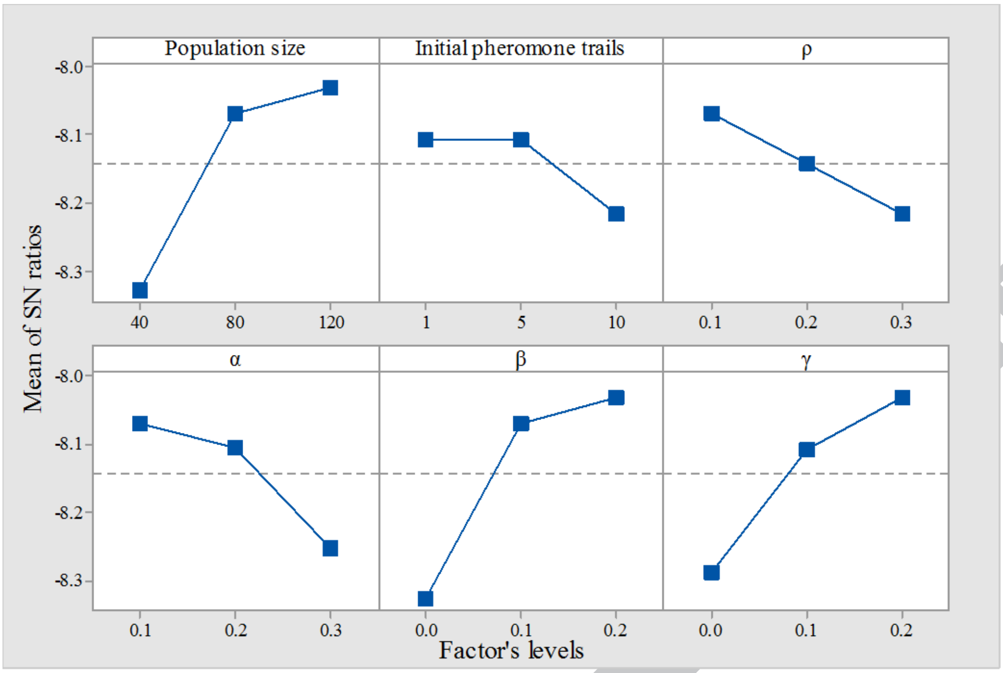


Fig. A2 The mean S/N ratio plot for each level of the parameters

Table A2 Parameter values of tested methods

Algorithm	Parameter	Tested values	Selected value
ACO1	Population size	40, 80, 120	120
	Initial pheromone trails	1, 5, 10	5
	Evaporation coefficient ρ	0.1, 0.2, 0.3	0.1
	α	0.1, 0.2, 0.3	0.1
	β	0.0, 0.1, 0.2	0.2
	γ	0.0, 0.1, 0.2	0.2
ACO2	Population size	40, 80, 120	120
	Initial pheromone trails	1, 5, 10	5
	Evaporation coefficient ρ	0.1, 0.2, 0.3	0.1
	α	0.1, 0.2, 0.3	0.1
	β	0.0, 0.1, 0.2	0.2
	γ	0.0, 0.1, 0.2	0.2
ACO3	Population size	40, 80, 120	120
	Initial pheromone trails	1, 5, 10	5
	Evaporation coefficient ρ	0.1, 0.2, 0.3	0.1
	α	0.1, 0.2, 0.3	0.1
	β	0.0, 0.1, 0.2	0.2
	γ	0.0, 0.1, 0.2	0.2
ACO4	Population size	40, 80, 120	80
	Initial pheromone trails	1, 5, 10	5
	Evaporation coefficient ρ	0.1, 0.2, 0.3	0.1
	α	0.1, 0.2, 0.3	0.1
	β	0.0, 0.1, 0.2	0.2
	γ	0.0, 0.1, 0.2	0.2
	LN	10, 100, 500	10 or 100
SACO	Population size	40, 80, 120	80
	Initial pheromone trails	1, 5, 10	5
	Evaporation coefficient ρ	0.1, 0.2, 0.3	0.1
	α	0.1, 0.2, 0.3	0.1

β	0.0, 0.1, 0.2	0.2
γ	0.0, 0.1, 0.2	0.2
LN	10, 100, 500	10 or 100
a	0.0, 0.005, 0.01	0.0
b	0.0, 0.1, 0.2	0.0
c	0.1, 0.2, 0.3	0.3

*Note: LN is set to 100 for problems with 83, 111, 148 and 297 tasks and 10 for other problems.

References

Aase, G.R., Olson, J.R. & Schniederjans, M.J. (2004). U-shaped assembly line layouts and their impact on labor productivity: An experimental study. European Journal of Operational Research, 156(3), 698-711.

Aase, G.R., Schniederjans, M.J. & Olson, J.R. (2003). U-OPT: An analysis of exact U-shaped line balancing procedures. International Journal of Production Research, 41(17), 4185-4210.

Aghay Kaboli, S.H., Selvaraj, J. & Rahim, N.A. (2017). Rain-fall optimization algorithm: A population based algorithm for solving constrained optimization problems. Journal of Computational Science, 19, 31-42.

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

Baykasoğlu, A. & Dereli, T. (2009). Simple and U-type Assembly Line Balancing by Using an Ant Colony Based Algorithm. Mathematical and Computational Applications, 14(1), 1.

Boysen, N. & Fliedner, M. (2008). A versatile algorithm for assembly line balancing. European Journal of Operational Research, 184(1), 39-56.

Celik, E., Kara, Y. & Atasagun, Y. (2014). A new approach for rebalancing of U-lines with stochastic task times using ant colony optimisation algorithm. International Journal of Production Research, 1-14.

Chiang, W.-C., Kouvelis, P. & Urban, T.L. (2007). Line balancing in a just-in-time production environment: balancing multiple U-lines. IIE Transactions, 39(4), 347-359.

Chiang, W.-C. & Urban, T.L. (2006). The stochastic U-line balancing problem: A heuristic procedure. European Journal of Operational Research, 175(3), 1767-1781.

De, A., Kumar, S.K., Gunasekaran, A. & Tiwari, M.K. (2017). Sustainable maritime inventory routing problem with time window constraints. Engineering Applications of Artificial Intelligence, 61, 77-95.

De, A., Mamanduru, V.K.R., Gunasekaran, A., Subramanian, N. & Tiwari, M.K. (2016). Composite particle algorithm for sustainable integrated dynamic ship routing and scheduling optimization. Computers & Industrial Engineering, 96, 201-215.

Dong, J., Zhang, L., Xiao, T. & Mao, H. (2014). Balancing and sequencing of stochastic mixed-model assembly U-lines to minimise the expectation of work overload time. International Journal of Production Research, 1-20.

Dorigo, M. & Blum, C. (2005). Ant colony optimization theory: A survey. Theoretical Computer Science, 344(2-3), 243-278.

Erel, E., Sabuncuoglu, I. & Aksu, B.A. (2001). Balancing of U-type assembly systems using simulated annealing. International Journal of Production Research, 39(13), 3003-3015.

Fattahi, A. & Turkay, M. (2015). On the MILP model for the U-shaped assembly line balancing problems. European Journal of Operational Research, 242(1), 343-346.

Gökçen, H. & Ağpak, K. (2006). A goal programming approach to simple U-line balancing problem. European Journal of Operational Research, 171(2), 577-585.

Gökçen, H., Ağpak, K., Gencer, C. & Kizilkaya, E. (2005). A shortest route formulation of simple U-type assembly line balancing problem. Applied Mathematical Modelling, 29(4), 373-380.

Hamzadayi, A. & Yildiz, G. (2012). A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints. Computers & Industrial

Engineering, 62(1), 206-215.

Hamzadayi, A. & Yildiz, G. (2013). A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. Computers & Industrial Engineering, 66(4), 1070-1084.

Hwang, R.K., Katayama, H. & Gen, M. (2008). U-shaped assembly line balancing problem with genetic algorithm. International Journal of Production Research, 46(16), 4637-4649.

Jayaswal, S. & Agarwal, P. (2014). Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach. Journal of Manufacturing Systems.

Kaboli, S.H.A., Selvaraj, J. & Rahim, N.A. (2016). Long-term electric energy consumption forecasting via artificial cooperative search algorithm. Energy, 115, Part 1, 857-871.

Kara, Y., Ozcan, U. & Peker, A. (2007). Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives. Applied Mathematics and Computation, 184(2), 566-588.

Kara, Y., Özgüven, C., Yalçın, N. & Atasagun, Y. (2011). Balancing straight and U-shaped assembly lines with resource dependent task times. International Journal of Production Research, 49(21), 6387-6405.

Kara, Y., Paksoy, T. & Chang, C.-T. (2009). Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. European Journal of Operational Research, 195(2), 335-347.

Kazemi, S.M., Ghodsi, R., Rabbani, M. & Tavakkoli-Moghaddam, R. (2011). A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. The International Journal of Advanced Manufacturing Technology, 55(9-12), 1111-1122.

Kim, Y.K., Kim, J.Y. & Kim, Y. (2006). An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. European Journal of Operational Research, 168(3), 838-852.

Kucukkoc, I. & Zhang, D.Z. (2015). Balancing of parallel U-shaped assembly lines. Computers and Operations Research, 64, 233–244, doi: http://dx.doi.org/210.1016/j.cor.2015.1005.1014.

Kucukkoc, I. & Zhang, D.Z. (2016). Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines. The International Journal of Advanced Manufacturing Technology, 82(1), 265–285.

Kucukkoc, I. & Zhang, D.Z. (2017). Balancing of mixed-model parallel U-shaped assembly lines considering model sequences. International Journal of Production Research, 1-18.

Li, Z., Tang, Q. & Zhang, L. (2017). Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study. Computers & Operations Research, 79, 78-93.

Maiyar, L.M. & Thakkar, J.J. (2017). A combined tactical and operational deterministic food grain transportation model: Particle swarm based optimization approach. Computers & Industrial Engineering, 110, 30-42.

Manavizadeh, N., Hosseini, N.-s., Rabbani, M. & Jolai, F. (2013). A Simulated Annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach. Computers & Industrial Engineering, 64(2), 669-685.

Miltenburg, G.J. & Wijngaard, J. (1994). The U-Line Line Balancing Problem. Management Science, 40(10), 1378-1388.

Modiri-Delshad, M., Aghay Kaboli, S.H., Taslimi-Renani, E. & Rahim, N.A. (2016). Backtracking search algorithm for solving economic dispatch problems with valve-point effects and multiple fuel options. Energy, 116, Part 1, 637-649.

Mogale, D.G., Kumar, S.K., Márquez, F.P.G. & Tiwari, M.K. (2017). Bulk wheat transportation and storage problem of public distribution system. Computers & Industrial Engineering, 104, 80-97.

Mozdgir, A., Mahdavi, I., Badeleh, I.S. & Solimanpur, M. (2013). Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. Mathematical and Computer Modelling, 57(1–2), 137-151.

Otto, A., Otto, C. & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. European Journal of Operational Research, 228(1), 33-45.

Özcan, U., Kellegöz, T. & Toklu, B. (2011). A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. International Journal of Production Research, 49(6), 1605-1626.

Pan, Q.-K., Gao, L., Li, X.-Y. & Gao, K.-Z. (2017). Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. Applied Mathematics and Computation, 303, 89-112.

Rabbani, M., Moghaddam, M. & Manavizadeh, N. (2012). Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout. International Journal of Advanced Manufacturing Technology, 59(9-12), 1191-1210.

Sabuncuoglu, I., Erel, E. & Alp, A. (2009). Ant colony optimization for the single model U-type assembly line balancing problem. International Journal of Production Economics, 120(2), 287-300.

Salveson, M.E. (1955). The assembly line balancing problem. Journal of Industrial Engineering, 6(3), 18-25.

Samà, M., D'Ariano, A., Corman, F. & Pacciarelli, D. (2017). Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas. Transportation Research Part C: Emerging Technologies, 80, 485-511.

Scholl, A. & Klein, R. (1999). ULINO: Optimally balancing U-shaped JIT assembly lines. International Journal of Production Research, 37(4), 721-736.

Tang, Q., Li, Z. & Zhang, L. (2016). An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-II. Computers & Industrial Engineering, 97, 146-156.

Toksarı, M.D., İşleyen, S.K., Güner, E. & Baykoç, Ö.F. (2008). Simple and U-type assembly line balancing problems with a learning effect. Applied Mathematical Modelling, 32(12), 2954-2961.

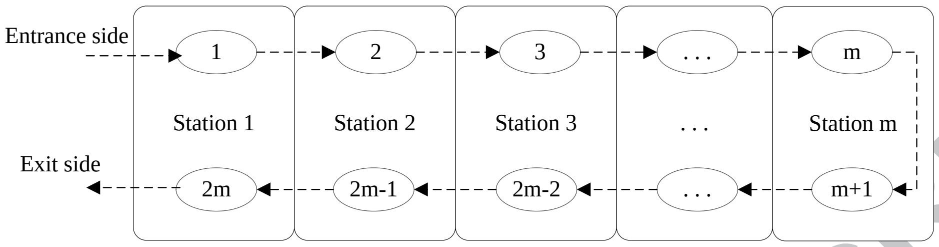
Urban, T.L. (1998). Note. Optimal Balancing of U-Shaped Assembly Lines. Management science, 44(5), 738-741.

Urban, T.L. & Chiang, W.-C. (2006). An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. European Journal of Operational Research, 168(3), 771-782.

Yegul, M.F., Agpak, K. & Yavuz, M. (2010). A New Algorithm for U-Shaped Two-Sided Assembly Line Balancing. Transactions of the Canadian Society for Mechanical Engineering, 34(2), 225-241.

Zheng, Q., Li, M., Li, Y. & Tang, Q. (2013). Station ant colony optimization for the type 2 assembly line balancing problem. The International Journal of Advanced Manufacturing Technology, 66(9), 1859-1870.

Graphical Abstract



Highlights

1. New MILP model is introduced and its validity is analyzed.
2. A new station-oriented ant colony optimization algorithm is developed.
3. Four MILP models are tested and evaluated.
4. The proposed station-oriented ACO method outperforms all the methods compared.
5. New best and optimum solutions are achieved for well-known benchmarks.