Sādhanā (2018)43:199
https://doi.org/10.1007/s12046-018-0969-9

© Indian Academy of Sciences

CrossMark

An ant colony optimisation algorithm for balancing two-sided U-type assembly lines with sequence-dependent set-up times

YILMAZ DELICE[1], EMEL KIZILKAYA AYDOĞAN[2,*], İSMET SÖYLEMEZ[3,4] and UĞUR ÖZCAN[4]

[1]Department of Management and Organization, Develi Vocational College, Erciyes University, 38400 Develi, Kayseri, Turkey
[2]Department of Industrial Engineering, Erciyes University, 38039 Talas, Kayseri, Turkey
[3]Department of Industrial Engineering, Abdullah Gül University, 38080 Kocasinan, Kayseri, Turkey
[4]Department of Industrial Engineering, Gazi University, 06570 Maltepe, Ankara, Turkey
e-mail: ekaydogan@erciyes.edu.tr; emelkizilkaya@gmail.com

MS received 1 March 2018; revised 31 May 2018; accepted 16 June 2018

Abstract. Some practical arrangements in assembly lines necessitate set-up times between consecutive tasks. To create more realistic models of operations, set-up times must be considered. In this study, a sequence-dependent set-up times approach for two-sided u-type assembly line (TUAL) structures is proposed for the first time. Previous studies on TUAL have not included set-up times in their analyses. Furthermore, an algorithm based on the Ant Colony Optimization (ACO) algorithm, which is using a heuristic priority rule based procedure has been proposed in order to solve this new approach. In this paper, we look at the sequence-dependent set-up times between consecutive tasks and consecutive cycles, called the "forward set-up time" and the "backward set-up time", respectively. Additionally, we examine the "crossover set-up time", which arises from a new sequence of tasks in a crossover station. In order to model more realistic assembly line configurations, it is necessary to include sequence-dependent set-up times when computing all of the operational times such as task starting times and finishing times as well as the total workstation time. In this study, the proposed approach aims to minimize the number of mated-stations as the primary objective and to minimize the number of total workstations as a secondary objective. In order to evaluate the efficiency of the proposed algorithm, a computational study is performed. As can be seen from the experimental results the proposed approach finds promising results for all literature-test problems.

Keywords. Assembly line balancing; U-type assembly lines; two-sided assembly lines; sequence-dependent set-up times; ant colony optimization; priority rules.

1. Introduction

An assembly line is a manufacturing system, in which a number of indivisible work elements (tasks) are consecutively performed on several productive units (stations) which are connected by some kind of transportation system such as a conveyor belt [1, 2]. Among the decision problems which arise in managing such systems, assembly line balancing problem (ALBP) is important one in medium-term production planning [3]. The first known formulation of the ALBP has been proposed by Salveson [4–7]. An ALBP is to obtain a feasible line balance which is defined as the assignment of assembly operations to a set of workstations in such a way that one or more objectives are optimized, with respect to precedence constraints and some specific restrictions of the assembly line system [6]. When the design structure is considered, various classifications can be made for the assembly lines based on the number of models produced in the assembly line (single, multi or mixed model), the flow type (U-type, straight), the nature of task times (probabilistic, deterministic), and station structure (one-sided, two-sided or multi manned) [8].

According to Kim *et al* [9], two-sided assembly lines are convenient to produce high-volume large-sized standardized products, such as automobiles, trucks and buses. According to Bartholdi [10], a two-sided assembly line has more advantages than one-sided one, i.e., the reduction of throughput time, the cost of tools and fixtures and number of operators. Both left-side and right-side of the assembly line are used in parallel in a two-sided assembly line, while only one side of the line is used in a one-sided assembly line [11]. Two-sided assembly lines have three different

*For correspondence
Published online: 01 November 2018

task types, that is to say either type tasks may be operated at either side, while *Right* or *Left* type ones are operated at one side of the assembly line.

Miltenburg and Wijngaard [12] presented the U-line balancing problem and Urban [13] developed the integer programming formulation of the U-line balancing problem. In U shaped lines the entrance and the exit stations of the line are formed at the same position. For this reason, operators are placed in the center of the U shape which allows stations with crossovers. U-lines can also provide several advantages over a straight assembly line. It is easier to adapt the fluctuations of demand thanks to the flexibility of increasing or decreasing the necessary number of workers [14]. Communication and visibility between operators are enhanced, and problem solving and the effort of adjusting to the changes are simplified [15].

There are limited studies on two-sided U-type assembly line balancing problem (TUALBP) in the literature. TUAL is a new assembly line structure that combines the advantages of both types of lines as underlined in Yegül *et al* [16], Ağpak *et al* [17] and Delice *et al* [18] and is developed for the production of large-sized products such as cars and trucks. A two sided assembly line model, in which one side of it arranged in U shape is presented by Yegül *et al* [16]. Ağpak *et al* [17] proposed a bi-objective 0-1 integer programming model for solving the TUALBP. Delice *et al* [18] developed a modified PSO algorithm to solve the TUALBP. Delice *et al* [19] developed a stochastic TUALBP and developed a genetic algorithm approach to solve it. Nevertheless, none of these studies on TUALBP has included sequence-dependent set-up times in their analyses. For this reason, the resulting models do not fully reflect the real life conditions.

Assembly line applications may require several practical arrangements, which may cause the set-up times depending on the sequence of consecutive tasks to be assigned to the same station.

In order to achieve more realistic assembly line structures, both the inter-station balancing and the intra-station scheduling of tasks must be considered simultaneously [20]. Balancing and scheduling tasks in assembly lines with sequence-dependent set-up times was first defined by Andrés *et al* [20]. They called the problem as the general assembly line balancing problem with setups (GALBPS). They solved simultaneously the inner-station balancing and the intra-station scheduling of tasks problems existence of sequence-dependent set-up times with low cycle times, and they designed eight different heuristic rules and a GRASP algorithm. Scholl *et al* [21] defined the concept of sequence-dependent task time increments and they formulated several versions of a mixed-integer program for sequence-dependent assembly line balancing problems. Martino and Pastor [22] proposed heuristic procedures, based on priority rules, for solving GALBPS, with set-ups. Özcan and Toklu [23] proposed a mixed integer program to sequence-dependent two-sided assembly lines, and they presented a heuristic approach (2-COMSOAL/S) to solve large-sized problems. Nazarian *et al* [24] presents mathematical models of manufacturing line design with the consideration of product change related inter-task times in evaluating station times for multi-model production. An optimization model is developed using mixed integer programming to minimize manufacturing line cost. Seyed-Alagheband *et al* [25] addressed the GALBPS of type-II. They proposed a mathematical model and a novel simulated annealing algorithm to solve it. Yolmeh and Kianfar [26] considered the set-up assembly line balancing and scheduling problem, involved task assignment and scheduling. They suggested a hybrid genetic algorithm that uses dynamic programming. Hamta *et al* [27] simultaneously considered minimizing the cycle time, minimizing the total equipment cost and minimizing the smoothness index objectives when task operation time is between the lower and upper bounds and sequence-dependent set-up times exist between the tasks. Akpinar *et al* [28] proposed a hybrid algorithm which executes ant colony optimization in combination with genetic algorithm for the mixed-model assembly line balancing problem with sequence-dependent set-up times. Scholl *et al* [29] modified the problem proposed by Andrés *et al* [20], more realistically, and gave a new and more compact mathematical model formulation and developed effective heuristic solution procedures. Akpinar and Baykasoğlu [30] considered mixed-model assembly line balancing problem with set-ups and developed a mixed-integer linear mathematical programming model. Akpinar and Baykasoğlu [31] considered a mixed-model assembly line balancing problem with sequence-dependent set-up times between tasks and developed a new multiple colony bees algorithm. Also, they proposed a neighbourhood approach based on the task selection strategy of the ACO. Esmaeilbeigi *et al* [32] present three new formulations for the set-up assembly line balancing and scheduling problem. Şahin and Kellegöz [33] was first defined the U-line balancing with sequence-dependent set-up times. They proposed a mathematical formulation, a simulated annealing approach and a genetic algorithm. The aim of their study is minimizing the cycle time for a given number of workstations. In their study, both forward set-ups and backward set-ups are considered, but crossover set-ups are not taken into account. Akpinar *et al* [34] describes an exact algorithm based on Benders decomposition to solve both simple and mixed-model assembly line balancing problems with sequence-dependent set-up times.

To the best of the authors' knowledge, there is no published study dealing with sequence-dependent set-up times on two-sided assembly lines with a layout in the form of U. For this purpose, in this study, two-sided U-type assembly line balancing problem with sequence-dependent set-up times (TUALBPS) is characterised in detail. In TUALBPS, "forward set-up time" and "backward set-up time" are discussed between consecutive tasks and consecutive cycles, respectively. Furthermore, "the crossover set-up

time", which arises from a new sequence of tasks in a crossover station, is proposed for the first time in this study. ALBP is NP-hard class of combinatorial optimization problems [35]. Different metaheuristics like GA, Ant Colony Optimization (ACO), Chemical Reaction Optimization (CRO) are used to solve the NP hard problems within reasonable computational time [36, 37]. An ant colony optimization (ACO) based algorithm is developed to solve the proposed problem. Although there are many different versions of the ACO algorithm, the classical ACO structure is preferred in this study in order to emphasize more on the sequence-dependent set-up time approach and allow future studies as well as benchmarks about the proposed model [38–41]. Exploring capability of the proposed ACO algorithm is reinforced by using a heuristic priority rule based procedure which is described in detail in section 3.

The remainder of the paper is organized as follows. TUALBPS is described in section 2. The proposed ACO is presented in section 3. The proposed algorithm is illustrated with a numerical example in section 4. The computational results of the proposed ACO on a set of test problems are presented in section 5. Finally, conclusions on this work and ideas for further research are presented in section 6.

2. Balancing two-sided U-type assembly lines with sequence-dependent set-up times

In the TUALBPS, a single-model two-sided U-type assembly line with sequence-dependent set-up times is considered. Two-sided U-type assembly lines consist of two parallel two-sided arms, i.e., front and back arm. Also, the entrance and the exit of these arms are formed at the same position. Each assembly operation is carried out at stations of positions, each of which has four locations on a U-shaped layout [18]. The tasks without any incomplete predecessors are performed at stations of the front arm, while the tasks without any incomplete successors are performed at stations of the back arm. Some of tasks from both of the central locations (locations 2 and 3 in figure 1) may be performed in a pair of two directly facing stations (crossover stations) by the same worker. In U-type lines, equal or less number of stations in comparison to the straight line is formed, owing to the crossover stations. Depending on the task side values (*L*-type, *R*-type and *E*-type) or precedence constraints of tasks, one crossover station, two distinct stations or empty station(s) can be occurred in the central locations, i.e., locations 2 and 3 in figure 1. Also, right or left side of the product can be worked on the central locations, depending on the entrance arms of the line. This causes to achieve more solutions. Consequently, both sides are considered and the solutions are generated for each direction, respectively, in order to achieve all possible solutions.

Considering potential sequence-dependent set-up times (e.g., travel times of operators, material movements and tool replacements) between consecutively performed tasks generates a more realistic assembly line structures. The GALBPS, defined by Andrés *et al* [20], considers the sequence-dependent set-up times not only between consecutive tasks in a station, but also between consecutive cycles. Scholl *et al* [29] modified and extended the approach of Andrés *et al* [20] by additionally distinguishing between forward and backward set-up times. Martino and Pastor [22] and Özcan and Toklu [23] assumed that the same set-up times are valid in both directions, i.e., $fs_{i,j} = bs_{i,j}$ for all task pairs i, j. Furthermore, they ignored the required time between consecutive cycles at a station which has on one assigned task, i.e., $fs_{i,i} = bs_{i,i} = 0$, for each task i. These assumptions substantially restrict the applicability of their models in practice. As can be seen at figure 1, a forward set-up time, $fs_{i,j}$, may occur between each consecutively assigned task pair at the same station. Furthermore, the backward set-up time, $bs_{i,j}$, may occur between the last task of the current cycle and the first task of the subsequent cycle at the same station. In this paper, in addition to the set-up times considered between consecutive tasks in a station (i.e., forward set-up time) and between consecutive cycles (i.e., backward set-up time), a new set-up time between consecutive tasks in a crossover station, namely, crossover set-up time is proposed. Two different crossover set-up times may occur in a crossover station. For instance, the first one occurs between the last task of the front arm of the station and the first task of the back arm of the station ($cs_{f,g}$ in figure 1). The second one occurs between the last task of the back arm of the station and the first task of the front arm of the station ($cs_{h,e}$. in figure 1).

In TUALBPS, the tasks are operated on a set of positions where there are maximum four operators working on opposite sides of the front and back arms of the assembly line, simultaneously. In a proposed model each precedence relationship between tasks is satisfied while ensuring to complete tasks within a predetermined cycle time (C). In figure 1, the general structure of the TUALBPS, forward, backward and crossover set-up times and unavoidable idle times can be seen in detail.

In this paper, our proposed algorithm uses three distinct matrices as set-up times, i.e., the forward set-up time matrix (*FSM*), the backward set-up time matrix (*BSM*) and the crossover set-up time matrix (*CSM*), to achieve a more realistic assembly line configuration. As we know from the literature, in two-sided U-type assembly lines, unavoidable idle times between some of the consecutive tasks may occur. For this reason, the sequence-dependent finishing time of each task must be considered carefully. Due to the compulsive precedence constraints, idle time (Δ), may occur in the TUALBPS, e.g., before the first assigned task (Δ_c, Δ_e and Δ_g), after the last assigned task (Δ_b and Δ_d) and between two consecutive tasks ($\Delta_{a,b}$, $\Delta_{e,f}$ and $\Delta_{g,h}$).

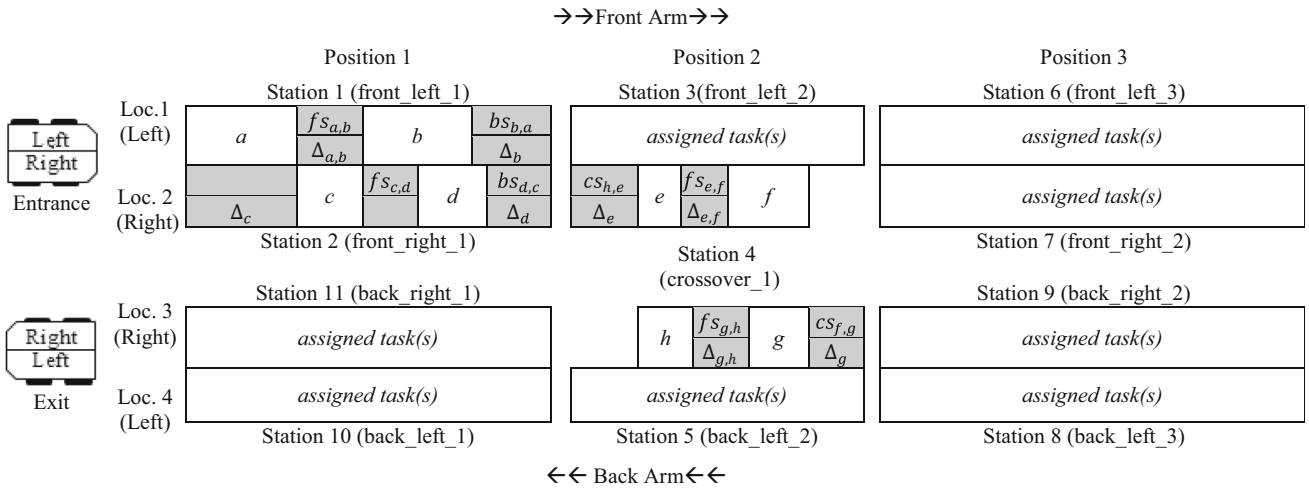

Figure 1. General structure of TUALBPS that workpiece enters the assembly line at front arm.

Figure 1 shows the detailed layout of Station 1, Station 2 and Crossover_Station1 (Station 4) of a single-model TUALBPS. In figure 1, each task number is placed at its relevant position inside the bars. The shaded rectangle shows all of the idle times (Δ), forward set-up times (fs), backward set-up times (bs) and crossover set-up times (cs).

In a TUALBPS, idle times (Δ) can be used in order to fulfil the set-up operations. For example, between task a and task b in Station 1, idle time $\Delta_{a,b}$occurs in order to satisfy the precedence relation between task c and task b (assume that task a is immediate predecessor of task b and c, and task c is immediate predecessor of task b and task d). Forward set-up time between task a and task b ($fs_{a,b}$) might be operated while $fs_{a,b} \leq \Delta_{a,b}$ condition satisfies. If the idle time, $\Delta_{a,b}$, is long enough to perform the set-up operations between task a and task b, then, no extra time is added to the equation of the finishing time of task b (ft_b). The finishing time of each task in Station 1 is calculated as follows:

- Task a: $ft_a = t_a$ ($\Delta_a = 0$)
- Task b: $ft_b = ft_a + max\{\Delta_{a,b}, fs_{a,b}\} + t_b$
- Task c: $ft_c = \Delta_c + t_c$
- Task d: $ft_d = ft_c + fs_{c,d} + t_d$

where t_i is operation time of task i. For both sides of the line the backward set-up values must also be taken into consideration to detect the feasibility ($F_{i,j}$) of the current line balancing operation, using Eq. (1) as follows:

$$F_{i,j} = \begin{cases} 1, & if\ C - \left(ft_{last} - ft_{first}\right) - t_{first} - bs_{last,first} \geq 0 \\ 0 & otherwise \end{cases}, \quad i = 1, 2, \ldots n,\ j = 1, 2 \tag{1}$$

For the left side of position 1 at front arm $F_{1,1}$ is calculated using the remainder time ($RT_{1,1}$) as follows:

–

$$RT_{1,1} = C - (ft_b - ft_a) - t_a - bs_{b,a}$$

For the right side of position 1, $F_{1,2}$ is calculated using the remainder time ($RT_{1,2}$) as follows:

–

$$RT_{1,2} = C - (ft_d - ft_c) - t_c - bs_{d,c}$$

When the values of each remainder times, ($RT_{1,1}$, $RT_{1,2}$), are greater than or equal to zero, the feasibility values of the stations become true ($F_{1,1} = 1$, $F_{1,2} = 1$).

Set-up times between the last task of the back arm and the first task of the front arm at the crossover station ($cs_{h,e}$) may be operated before the first task of the front arm or after the last task of the back arm, according to the assignment sequence of the tasks. In order to focus to the crossover set-up time approach and its basic features, layout of the Station 3 and Station 5 and other precedence relations between tasks are not considered in figure 1. In the case of the crossover set-up, between task h and e ($cs_{h,e}$), is operated before the first task of the front arm (task operation sequence of assembly process is assumed to be e, f, g and h), the finishing time of each task in Station 4 is calculated as follows:

- Task e: $ft_e = max\{\Delta_e, cs_{h,e}\} + t_e$
- Task f: $ft_f = ft_e + max\{\Delta_{e,f}, fs_{e,f}\} + t_f$
- Task g: $ft_g = ft_f + max\{\Delta_g, cs_{f,g}\} + t_g$
- Task h: $ft_h = ft_g + max\{\Delta_{g,h}, fs_{g,h}\} + t_h$

$\Delta_{g,h}$ and $fs_{g,h}$ are used to calculate finishing time of task g, because the actual operation direction of the back arm is right to the left. For the crossover station, the crossover set-up values between the last task of the front arm and the first

task of the back arm must also be taken into consideration to detect the feasibility of the current line balancing operation.

The basic assumptions of TUALBPS are as follows:

- Each task is performed, simultaneously, at both sides of the U shaped line.
- Deterministic task times and set-up times are used.
- Set-up times include walking times of operators, material movements and tool replacements.
- Some of the tasks must be operated at one-side and others may be operated at either side.
- The precedence relationships among tasks are known and a single model of a product is produced.
- Equally equipped stations are used; each task can only be assigned to one station.
- Work-in-process inventory is not allowed.
- Parallel tasks and parallel stations are not allowed.

3. The proposed algorithm: an ant colony approach

Ant colony optimization (ACO) is one of the population-based metaheuristic which mimics the collective capability of real ant colonies to find the shortest route between the nest and a food source. Ants utilize a special chemical substance called pheromone to communicate and exchange the knowledge between colony members. The amount of the substance in each route is reduced by time because of evaporation effect. Furthermore, the pheromone of a route is increased by those ants that passed through that particular route. With more ants following the same route, its pheromone deposit, left behind on the route, accumulates faster. Given a choice of many routes, an ant would choose the route with a higher pheromone concentration using the sense of smell. Hence, the amount of pheromone in a route is controlled by two factors, the rate of evaporation and the number of ants who passed through that particular route. The first ant algorithm, Ant System, is proposed by Colorni *et al* [42, 43] and used to solve the travelling salesman problem. Bautista and Pereira [44, 45], McMullen and Tarasewich [46] and Sabuncuoglu *et al* [47] presented several heuristics for the ALBP using concepts derived from ACO algorithm. Simaria and Vilarinho [48] and Yagmahan [49] proposed new ACO techniques. Akpinar *et al* [28] hybridized genetic algorithm with an ACO algorithm. Kucukkoc and Zhang [50] proposed a flexible agent-based ACO approach for the mixed model parallel two-sided ALBP. Blum proposed BeamACO, the combination of ACO algorithms with beam-search [38]. Ding *etal* [39] proposed a hybrid ant colony optimization (HACO). Mogale *etal* [40] proposed an effective metaheuristic which based on the strategy of sorting elite ants and pheromone trail updating called Improved Max-Min Ant System (IMMAS). Dorigo and Stützle [41] observed many developments in ACO and gained an overview of recent research trends in ACO.

In this study, an ACO algorithm based solution approach is developed to solve the TUALBPS. Each ant (solution) of the proposed ACO algorithm aims to find good solution for the TUALBPS. Each member of the colony uses pheromone trails matrix and the heuristic information matrix, in order to achieve the feasible solution. The general structure and pseudo code of the proposed ACO algorithm are shown in figures 2 and 3, respectively.

3.1 *Initialization of the matrices*

Initialization step consists of initialization of each basic matrix and parameter, used by the proposed ACO in order to solve the TUALBPS. Our proposed algorithm uses several matrices. Some of them use constant values while others use variable values. All of the basic matrices are listed below:

– *Precedence matrix* ($PM_{i,j} i, j = 1, 2, \ldots, n$) keeps the all precedence relations between tasks. If there is a precedence relation between task i and task j, the value of PM[i, j] is set to 1, otherwise 0. The *PM* matrix is used to determine the candidate tasks list (*CL*) for the assignment operation.
– *Task matrix* ($TM_{i,m} i = 1, 2, \ldots, n$and $m = 1, 2, 3$) keeps task number, task side and task time values of each problem.
– *Forward set-up matrix* ($FSM_{i,j} i, j = 1, 2, \ldots, n$) keeps the forward set-up times between all tasks. The forward set-up values are used whenever a task j is performed next to task i at the same station, in order to compute the global operation time.
– *Backward set-up matrix* ($BSM_{i,j} i, j = 1, 2, \ldots, n$) keeps the backward set-up times between all tasks. If task i is the last task in a station in which task j was the first task in the same station, the backward set-up values between task i and task j are used in order to compute the global operation time.
– *Crossover set-up matrix* ($CSM_{i,j} i, j = 1, 2, \ldots, n$) keeps the crossover set-up times between all tasks. If task i is the last task in the front arm/back arm of the crossover station in which task j is the first task in the back arm/front arm of the same crossover station, the crossover set-up value between task i and task j is used in order to compute the global operation time of the crossover station.
– *Pheromone matrix* ($\tau_{t,i} t, i = 1, 2, \ldots, n$) saves the pheromone trail intensity of the task i stored in the t^{th} task assignment process. These values are required to calculate the selection probability (P) of each task at the assignment process.

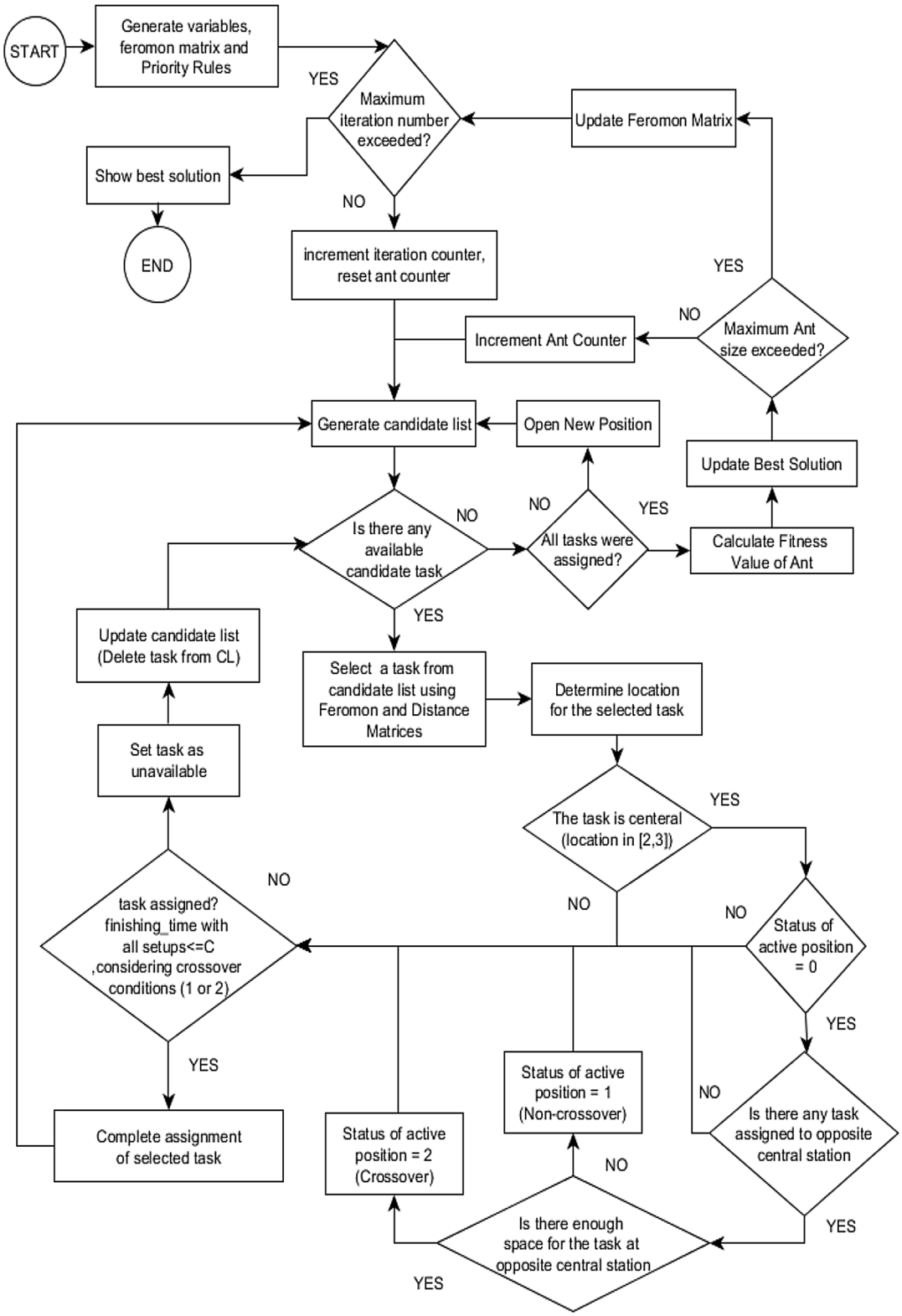

Figure 2. General structure of the proposed ACO algorithm.

– *Heuristic information matrix* ($\eta_{a,t} a = 1, 2, \ldots, CS$ and $t = 1, 2, \ldots, n+1$) is used to save one of the six different heuristic information which is required to calculate the selection probability (P) of t^{th} task assignment process for the ant a. The last index of the η matrix saves the side selection type value for the assignment of E-type

Step1. Initialize Step
  Step1.1. Set *iter* = 1, *a*= 1
Step 2. Solution Step
  Do while *iter*≤*IS* (*iteration_size)*
    Do while *a*≤*CS (colony_size)*
      Step 2.1. Generating new colony
      Step 2.2. Evaluating objective functions
      Step 2.3. Updating best solution
      Step 2.4. Updating Feromon Matrix
      Increment *ant_number*(*a*)
    end do
  Increment *iteration_number*(*iter*)
  end do
Step 3. Show best solution
Step 4. Finish

Figure 3. Pseudo code of the proposed ACO algorithm.

tasks. In the first iteration, η matrix is generated with random integer numbers between [17, 34]. The last index is assigned using binary values (0 for the side which has more available time, 1 for the random selection).

To evaluate the quality of each candidate ant, the solution matrix is used in the proposed algorithm in order to save the detailed layout and the objective function values for each ant.

3.2 *Candidate list and priority rules*

After the initialization step, the algorithm becomes ready for the assignment procedure. In order to perform the task assignment process of the proposed algorithm, all of the assignable tasks are combined into a candidate list (*CL*). Some of the tasks whose predecessors have already been assigned are defined as Front-type while some tasks whose successors have already been assigned are defined as back-type. All front-type and back-type tasks combined in *CL*, according to the precedence diagram. A task is selected from the candidate list using the probability value (P_i) of each candidate task by the roulette wheel selection strategy. For the current ant, the selection probability value of each task is calculated using ant's pheromone value and the selected priority rule, using Eqs. (2)–(4).

$$pr_i = \frac{X_{r,i}}{\sum X_{r,i}}, r = \eta_{a,t} \text{ and } i \in CL \tag{2}$$

where $\eta_{a,t}$ has the priority rule value of the t^{th} task assignment operation of ant a. The $X_{r,i}$ matix is used to save all of the priority rule values for each task i in *CL* and all values are determined in the initialization step. Vector pr_i is used to save all of the calculated relative priority rule values according to the selected priority rule type r.

$$P_i = \frac{\alpha * \tau_{t,i} + \beta * pr_i}{\sum \alpha * \tau_{t,i} + \beta * pr_i}, \; i \in CL \text{ and } t \text{ is the current assignment number} \tag{3}$$

where α and β are the parameters which determine the relative importance of pheromone matrix versus heuristic information matrix. According to the cumulative selection probability matrix (*SP*) and the randomly generated q value ($q \in (0,1)$), a task is chosen from the candidate list, randomly as follows:

$$SP_l = SP_{l-1} + P_i, l = 1, \ldots, nc(nc\,is\,\#\,of\,cl), \; i \in cl, \text{ where } (SP_1 = 0, SP_{nc} = 1) \tag{4}$$

The task, whose cumulative probability value satisfies $SP_{l-1} \le q < SP_l$ rule, is chosen for the assignment process. Then assignment procedure is implemented.

Bautista *et al* [51] firstly proposed the priority rule based approach about assignment of tasks to the workstations. In the literature, some priority rules have been proposed. Helgeson and Birnie [52] proposed maximum ranked positional weight, Tonge [53] proposed maximum number of immediate followers, Kilbridge and Wester [54] proposed maximum task time, Arcus [55] proposed random task assignment, Moodie and Young [56] proposed maximum task time first, Brian and Patterson [57] proposed maximum total number of follower tasks, Elsayed and Boucher [58] proposed minimum total number of predecessor tasks and minimum reverse positional weight and also Talbot *et al* [59], Scholl and VoB [60] and Boctor [61]. The six different priority rules used to calculate the probability value of each task in the proposed ACO algorithm are as follows:

- The task number: Select task with smallest task number,
- The total number of successor tasks: Select task having most followers,
- The total number of predecessor tasks: Select task having most predecessors,
- The ranked positional weight: Select task with highest ranked positional weight,
- The processing time: Select task with longest duration,
- Random assignment: Select task randomly.

3.3 *Task assignment procedure*

The task assignment procedure is executed, repeatedly, for each ant of the existing colony during whole iterations. At first step of the procedure an empty position with four locations is opened. Depending on the arm of the line that the assembly process start, different alternative line balances may be achieved. That is to say, if the assembly

process start at front arm, center of the two-sided U-line becomes right (R), otherwise center of the two-sided U-line becomes left (L). And this enlarges the solution space of the problem and may produce different possible solutions.

After determining operation side of the line, a candidate task is selected by the selection procedure from the *CL*. The location of the current position should be determined in order to apply the assignment procedure. It is determined depending on the side value of the selected task, i.e., *L*-type, *R*-type and *E*-type tasks, selection side of precedence diagram value of the task (*front* or *back*) and the center of the line value (R or L). For the *L*-type and *R*-type tasks, the assignment location is already known. That is to say, if the selection side of the precedence diagram value is *front* and the center of the assembly line is R/L, the *L*-type task is assigned to location 1/location 2 while the *R*-type task is assigned to location 2/location 1. Similarly, if the selection side of the precedence diagram value is *back* and the center of the assembly line is R/L, the *L*-type task is assigned to location 4/location 3 while the *R*-type task is assigned to location 3/location 4. However, for the *E*-type tasks, the operation side is selected using the side type value of the active ant. The last index of the each heuristic information matrix of the current ant determines the assignment side for the *E*-type tasks. For example, the side which has more available time is selected when the last index value of the heuristic information matrix is 0 while the random side selection is occurred when the value is 1. To select the side, which has more available time, for the *E*-type tasks, provides better utilization and less idle times but it may also prevent to reach some of the possible solutions. Therefore, it is also preferred to use the random side selection method for the *E*-type tasks to make the proposed algorithm having ability of finding all possible solutions.

After the task number and the position value are determined, the task assignment operation is completed by assigning the task information to the solution matrix. Then, the solution and solution results are determined by calculating the sequence-dependent starting and finishing times including forward, backward and crossover set-up times for all locations of the line. At each assignment process, the crossover availability of the current position must be controlled. The crossover situation for the empty central locations (locations 2 and 3) may have three different values, i.e., 0 (undefined), 1 (non-crossover) or 2 (crossover) as can be seen in figure 2. At first, the crossover situation value is set as undefined for each location. At the assignment process, if some of the tasks are assigned to stations at locations 2 and 3, concurrently, the crossover station occurs and the crossover situation is set as two. Sometimes, stations at location 2 and 3 are filled sequentially (one after the other). Depending on task number, task side and the center of the line value some of the selected tasks are assigned to only a station at location 2/location 3 while the station at location 3/location 2 is empty, the crossover station cannot occur and the crossover situation is set as one.

3.4 *Evaluating objective function*

After computing the solutions of the proposed ACO at each step, the solution qualities are evaluated by considering the objective function values. Performance level of each candidate solution is determined by objective function values. End of each step of ACO algorithm, the pheromone information matrix is influenced by the best solution, in other words, best solution influences the new ant colonies which will be generated at next iterations.

In the proposed ACO for TUALBPS, the primary objective is minimizing the number of positions and the secondary objective is minimizing the number of total stations, for a predetermined cycle time. Since the *NP* is more important than *NS*, it is multiplied by a sufficiently large number(φ). Therefore, Eq. (5) is used to compute the objective function value of each solution as follows:

$$Min\ Z = \varphi \cdot NP + NS \tag{5}$$

3.5 *Updating the best solution and pheromone matrix*

In every step of the algorithm, the ant which has the minimum total objective function value, so far, is defined as the best candidate solution. The best solution is updated at the end of the each assignment process during the whole iterations. Pheromone updating is done to avoid premature search convergence and to add the ants' search experience that contains good or promising solutions into the pheromone structure. Pheromone updating procedure is done by decreasing all the pheromone values through pheromone evaporation and increasing the pheromone values associated with a chosen set of good solution. Updating of the pheromone matrix is applied by evaporating of existing pheromone quantity using Eq. (6).

$$\tau_{t,i} = (1 - \rho) * \tau_{t,i} \tag{6}$$

where ρ a coefficient is called the *evaporation rate*. At the end of the each iteration, pheromone matrix is updated according to the following formula:

$$\tau_{t,i}(iter + 1) = \tau_{t,i}(iter) + \Delta\tau_{t,i}^{best} \tag{7}$$

where $\Delta\tau_{t,i}^{best} =$

$$\begin{cases} 0.01, & \text{if the best ant so far uses } (t,i) \text{ assignment} \\ 0, & \text{otherwise} \end{cases}$$

In addition to the pheromone update heuristic information matrix is also updated in order to avoid premature search convergence. Also, a local search mechanism is added to the updating process to improve the solutions

obtained by the best ant. At the end of the each iteration, heuristic rule matrix is updated using two-stage process. At the first stage of the updating process of the heuristic information matrix, a local search is applied for the first 20% part of the ants, using the best ant achieved so far. For this purpose, two index numbers are selected, randomly, and a swap mechanism is applied to the heuristic information values of the best ant solution using these two index numbers. The heuristic rule numbers at these two index numbers are switched and achieved values are saved to complete the updating mechanism. At the second stage of updating procedure of the heuristic information matrix, a random heuristic rule assignment is applied for the other part of ants, to complete the updating mechanism.

3.6 *Algorithm parameters*

In this study, the basic ACO parameters used in the proposed ACO are colony size = 100, number of iterations = 1000, $\alpha = 0.1$, $\beta = 0.1$ and evaporation rate = 0.01. They are obtained from preliminary experiments.

4. An illustrative example

A numerical example is used to show the important characteristics of the proposed algorithm for TUALBPS. The example problem has the following characteristics:

- A single model product is assembled in a two-sided U-type assembly line with the cycle time of $C = 8$.
- Task times, task sides and the precedence diagram of 12 tasks are given in figure 4.
- Table 1 shows forward, backward and crossover set-up times of the tasks, respectively.

Representation of an assembly line balancing solution built by the proposed ACO for high set-up times (Center is Right) is presented by figure 5. It shows the sequence and the operational times of each task. According to figure 5,

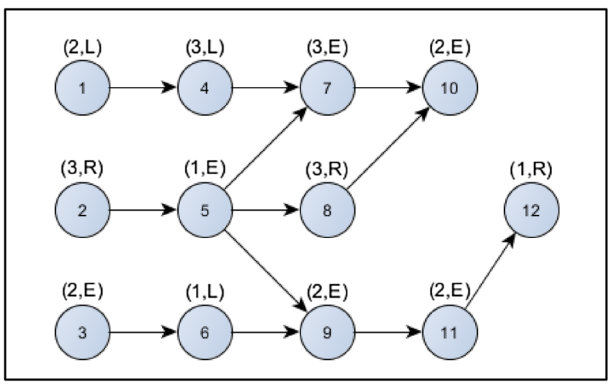

Figure 4. An example problem (P12).

Table 1. Set-up time matrices (high variability).

Tasks	Forward set-up times												Backward set-up times												Crossover set-up times											
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	1	0	0	0	1	1	2	0	0	0	1	0	1	0	2	0	1	1	1	0	0	0	0	0	1	0	1	0	2	2	1	0
2	0	0	2	0	2	0	0	2	0	2	1	1	0	0	2	0	1	0	2	1	1	1	1	0	0	0	2	0	1	0	1	0	0	2	2	1
3	2	0	0	1	0	1	1	1	1	1	2	1	0	0	0	0	1	1	2	1	1	1	1	0	0	0	0	0	1	2	0	1	1	2	1	1
4	0	0	0	0	2	2	2	0	2	2	2	0	0	0	1	2	2	2	1	0	1	1	2	0	0	0	1	0	0	0	0	0	1	1	0	0
5	1	1	1	2	0	1	1	1	0	1	1	0	0	0	2	1	0	0	1	0	0	2	1	0	0	0	1	0	1	1	0	0	0	0	2	0
6	0	0	2	1	2	0	1	0	1	0	2	0	0	0	2	1	1	0	2	0	1	1	0	0	0	0	2	0	1	0	1	1	0	1	1	0
7	1	0	1	2	1	1	0	1	1	1	1	2	0	0	0	2	2	0	1	1	0	1	2	0	1	0	0	0	2	0	0	0	1	0	2	1
8	0	1	0	0	0	0	2	0	1	1	0	0	0	0	1	2	0	0	1	1	2	1	0	1	0	0	1	0	2	1	2	1	1	0	0	0
9	2	1	2	1	0	1	1	2	0	1	0	2	0	0	2	2	1	2	2	0	1	1	1	0	0	0	1	0	1	1	0	2	0	0	1	1
10	2	1	0	0	1	1	1	1	1	0	1	1	0	0	0	2	2	0	1	2	1	1	2	1	0	0	0	0	1	0	1	1	0	0	1	0
11	2	0	2	1	1	1	1	1	1	1	0	2	0	0	2	2	1	2	2	1	1	1	1	0	0	0	1	0	2	1	2	1	2	1	0	0
12	0	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0	1	0	2	0	1	1	1	0	0	0	1	0	1	0	0	2	1	0	2	0

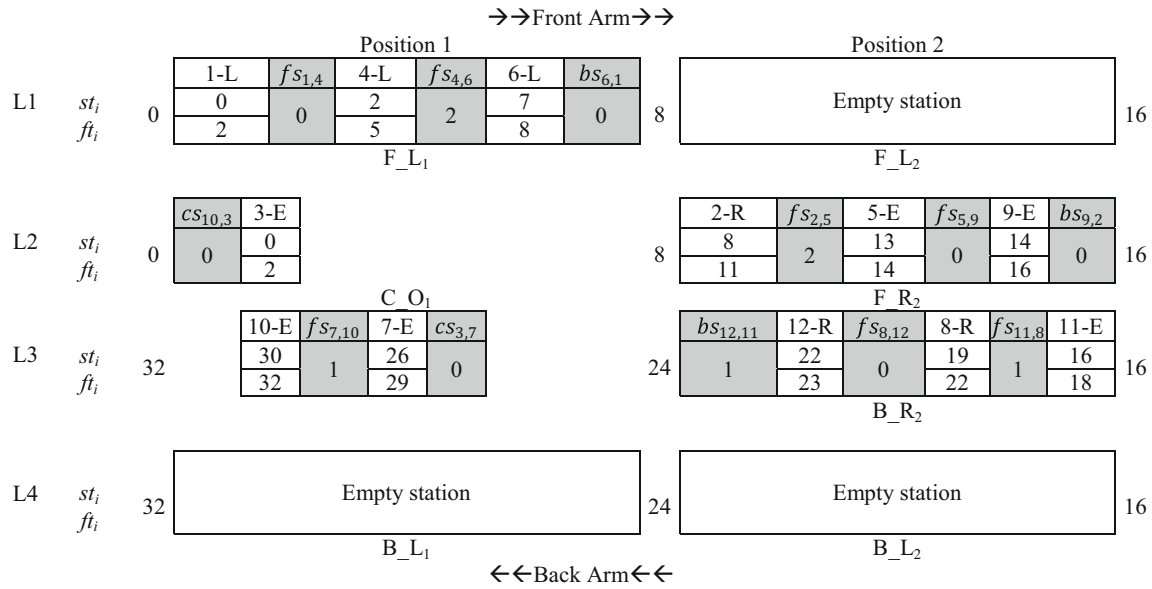

Figure 5. Representation of an assembly line balancing solution for high setup times (Center is Right).

the TUALBPS model has four different locations with two positions, i.e., the front left (F_L), front right (F_R), back right (B_R) and back left (B_L). In addition to the three non-crossover stations (F_L_1 and F_R_2 in front arm and B_R_2 in back arm) and three empty stations (F_L_2 in front arm and B_L_1 and $B_L_2$in back arm), a crossover station is arranged in the solution layout, i.e., C_O_1. Total number of used stations is four. In addition to generating layout which requires less set-up times, these crossover stations also reinforce the proposed algorithm to achieve better layout solutions which require fewer operators.

All of the stations, arranged at 4 locations of current position, are identified using location name and sequential numbers. Allocations of each task into the stations are performed depend on the sequence-dependent finishing time and cycle time values. The task number, location, start time (st_i), finish time (ft_i) and station number values of each task and forward, backward and crossover set-up times between tasks are shown in figure 5. An arrow mark shows the operational direction of the assembly line. The operation direction is left to right at front arm while it is right to left at back arm.

5. Computational study

Literature test problems have been used to demonstrate the effectiveness of the proposed ACO algorithm. Solution quality and algorithm performance are compared with best known results for two-sided U-type assembly line balancing problem without set-up times [18]. Four of test problems are small-sized (P9, P12, P16 and P24) and three of them are large-sized (P65, P148 and P205). P16, P65 and P205 are taken from Lee *et al* [11], P9, P12 and P24 are taken from Kim *et al* [9] and P148 is taken from Bartholdi [10] and modified by Lee *et al* [11]. Two different cycle time are considered for P9 problem and others as follows: P12(4), P16(3), P24(5), P65(5), P148(6) and P205(10). This means that 14 small-sized and 21 large-sized instances are evaluated. The precedence relationships, the operational directions and the task times are not changed. In the paper, new data sets including forward set-ups, backward set-ups and crossover set-ups are generated with considering Scholl *et al* [29] data set generation concept.

The proposed ACO approach is coded in Borland Delphi 7. Each test problem is tested on a computer which has Intel Xeon(R) CPU E-5-2695, 2.4 GHz processor and 32 GB RAM. ACO algorithm is run ten times for each cycle time of each test problem. The number of positions, number of stations and average CPU time values are presented in table 2 for the two possible situations that the item enters the assembly line at front arm (L) or back arm (R). Table 2 summarizes the results of all test problems with low set-up time variability level and high set-up time variability level l for both of the solutions, left (L) and right (R). According to table 2 and figure 6(a), the computational times for the test problems with low set-up variability level are less than one second at 12 of 35 instances. Results of 11 of 35 instances are obtained less than one second for high set-up variability. And also, all instances are calculated not more than 3000 seconds. Therefore, computational analysis is not required for these problem sets. Computational time variabilities are depended on both low and high set-up variability levels as given in figure 6(a). The main inference of the figure 6(a) is solution times of the test problems with low set-up variability level less than the test problems with

Table 2. Computational results of proposed ACO.

				Low set-up variability level						High set-up variability level					
		Without set-up (Delice *et al* 2017)		NM[NS] (min)		NM[NS] (avg)		CPU time (s)		NM[NS] (min)		NM[NS] (avg)		CPU time (s)	
Problem	C	R	L	R	L	R	L	R	L	R	L	R	L	R	L
P9	5	4	4	1[4]	1[4]	1[4]	1[4]	< 0.01	< 0.01	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01
	6	3	3	1[4]	1[4]	1[4]	1[4]	< 0.01	< 0.01	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01
P12	5	5	5	2[6]	2[6]	2[6]	2[6]	< 0.01	< 0.01	3[7]	3[7]	3[7.1]	3[7]	< 0.01	1
	6	5	5	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01	2[6]	2[6]	2[6]	2[6]	< 0.01	< 0.01
	7	4	4	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01
	8	4	4	1[4]	1[4]	1[4]	1[4]	< 0.01	< 0.01	2[**4**]	2[5]	2[4.5]	2[5]	< 0.01	< 0.01
P16	16	6	6	2[7]	2[7]	2.7[7]	2[7]	< 0.01	< 0.01	3[8]	3[8]	3[8]	3[8]	1	1
	19	5	5	2[6]	2[6]	2[6]	2[6]	< 0.01	< 0.01	2[6]	2[6]	2[6.5]	2[6.4]	1	1
	22	4	4	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01	2[6]	2[6]	2[6]	2[6]	< 0.01	< 0.01
P24	20	7	7	2[8]	2[8]	2.9[8.1]	2.9[8.2]	3	3	3[9]	3[9]	3[9.4]	3[9.5]	2	2
	25	6	6	2[7]	2[7]	2[7]	2[7]	< 0.01	< 0.01	2[8]	2[8]	2[8]	2[8]	< 0.01	1
	30	5	5	2[6]	2[6]	2[6]	2[6]	< 0.01	< 0.01	2[6]	2[6]	2[6.5]	2[6.1]	10	6
	35	4	4	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01	2[5]	2[5]	2[5.6]	2[5.6]	13	23
	40	4	4	1[4]	1[4]	1.5[4.1]	1.8[4.3]	16	17	2[5]	2[5]	2[5]	2[5]	< 0.01	< 0.01
P65	326	17	17	6[19]	6[19]	6[19]	6[19]	5	8	6[21]	6[21]	6[21.9]	6.2[21.8]	21	19
	381	14	14	5[16]	5[16]	5[16]	5[16.2]	18	14	5[18]	5[18]	5.3[19.1]	5.3[18.9]	21	25
	435	12	12	4[14]	4[14]	4.2[14]	4.5[14.3]	55	8	5[16]	5[16]	5[16]	5[16.1]	7	9
	490	11	11	4[13]	3[**12**]	4[13]	3.9[12.8]	16	360	4[14]	4[14]	4[14.9]	4.3[14.6]	18	11
	544	10	10	3[11]	4[11]	3.6[11.2]	4[11.4]	52	16	4[13]	4[13]	4.2[13]	4.1[13]	2	3
P148	255	21	21	7[24]	7[24]	7[24]	7[24.3]	142	178	7[24]	6[24]	7[25.5]	7.1[25.4]	494	845
	306	17	17	6[20]	6[20]	6[20.1]	6[20.6]	81	123	6[20]	6[20]	6[20.9]	6.2[21.6]	510	835
	357	15	15	5[17]	5[17]	5[17.7]	5.4[18]	125	149	5[17]	5[17]	5.4[18.6]	5.9[18.3]	472	1187
	408	13	13	5[15]	5[15]	5[15]	5[15.6]	125	137	5[15]	4[15]	5[15.8]	4.9[16.2]	107	1210
	459	12	12	4[13]	4[13]	4[13.9]	4.4[14.1]	460	99	4[14]	4[14]	4.4[14.4]	4.8[14.9]	101	166
	510	11	11	4[12]	4[12]	4[12]	4[12.2]	128	121	4[12]	4[12]	4[12.9]	4[12.9]	485	1152
P205	1133	22	22	8[25]	8[25]	8[26.7]	8[25.8]	882	150	8[26]	8[26]	8.9[27.3]	8[27.2]	112	150
	1322	19	19	7[22]	7[**21**]	7[22.8]	7[21.8]	153	99	7[23]	7[**22**]	7.2[23.5]	7[23.1]	130	526
	1510	17	17	6[20]	6[**19**]	6[21.1]	6[19.9]	137	129	6[20]	6[20]	6.5[21.1]	6[20.6]	441	139
	1699	15	15	6[17]	6[17]	6[17.9]	6[17]	409	140	6[18]	6[**17**]	6[18.4]	6[17.7]	128	136
	1888	13	13	5[16]	5[**15**]	5[17]	5[15.8]	648	129	5[17]	5[**16**]	5.4[17.3]	5 [16.3]	134	159
	2077	12	12	5[14]	5[14]	5[14.9]	5[14.3]	133	124	5[15]	5[**14**]	5[15.5]	5[14.9]	143	177
	2266	11	11	4[13]	4[13]	4.8[13.7]	4[13.9]	929	930	4[14]	4[14]	4.8[14.1]	4 [14.6]	209	161
	2454	10	10	4[12]	4[12]	4[13.2]	4[12.6]	1159	124	3[12]	4[12]	3.9[12.9]	4 [12.8]	2907	132
	2643	10	10	4[11]	4[11]	4[12.1]	4[11.9]	1757	2561	3[11]	4[11]	3.9[11.9]	4[11.9]	2549	533
	2832	9	9	4[11]	3[11]	4[11.7]	3.1[11.8]	192	181	4[11]	3[11]	4[11.4]	3.5[11.9]	176	1220

Bold characters are used to indicate the solutions that achieve better results from the opposite side of the assembly line.

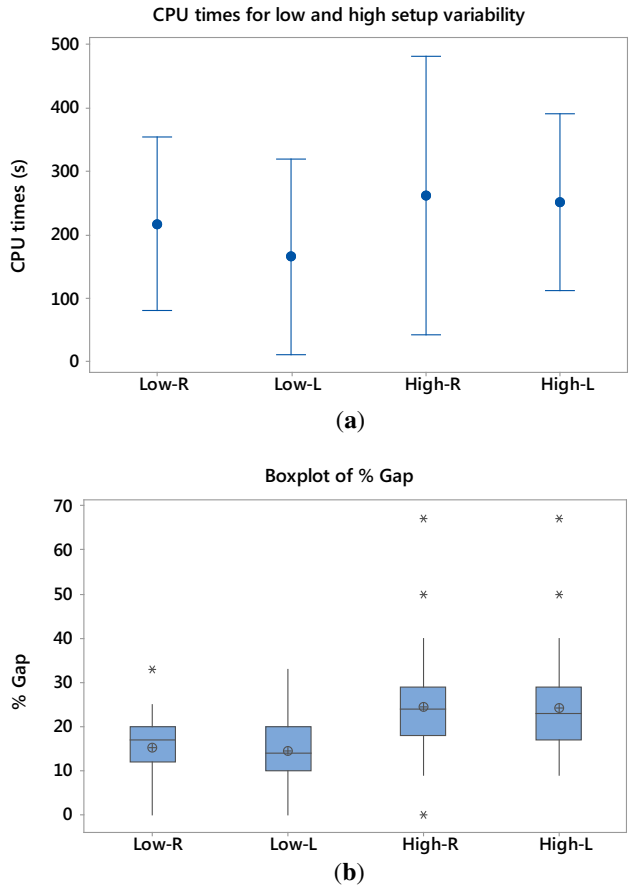

Figure 6. The results of the proposed ACO approach (**a**) CPU times and (**b**) boxplot for the Gap%.

high set-up variability level. Another inference is that R solutions have bigger range than L for the test problems with high set-up variability level.

Box plot of deviation from the U-type two sided assembly line balancing problem without set-up times results are given in figure 6(b) for the proposed ACO. Although there are some outliers, most results have acceptable deviations. Because 29 and 30 of 35 instances have less/equal 20% Gap for R and L at low set-up variability, respectively. High set-up variability results are not as good as low's but 15 and 16 of 35 instances are not exceeded 20% Gap for R and L, respectively.

Summary results of the test problems according to the problem size are shown in table 3. In addition, number of instances for each size of problems and average percent deviation values are given. As seen in table 3, proposed approach has less than 20% Gap except P16. Especially, left side entrance direction results are more acceptable than right side for the large sized test problems as P65 and P205.

Deviation of the number of station from the best known without set-up times results are given in table 4. There is no deviation for low set-up variability level at both R and L entrance direction for 4 of 35 instances. Moreover, 14 of 35 problems have only deviation of only one station. This means

Table 3. Results of the test problems according to the problem size.

		Proposed ACO Gap%	
Problem types	Number of instances	R	L
P9	2	16.66	16.66
P12	4	11.25	11.25
P16	3	20.55	20.55
P24	5	15.19	15.19
P65	5	14.18	12.36
P148	6	13.01	13.01
P205	10	17.05	15.17

Table 4. Deviation of the number of station.

	Proposed ACO			
	Low set-up variability		High set-up variability	
Number of station deviation	R	L	R	L
0	4	4	1	0
1	13	14	9	10
2	11	13	11	13
3	4	4	8	7
4	3	0	3	5
Mean	1.69	1.49	2.26	2.2

that more than half of the instances have acceptable deviation of number of station which is less and equal than one station. However, the proposed approach is not as efficient as for high set-up variability. Number of acceptable deviation instances for R and L entrance side is 10 instances. And also Gap% of means is shown for both low and high set-up time variability level. While deviations of means are 1.69 and 1.49 for low set-up variability, 2.26 and 2.2 are for high set-up variability level. Lastly, all results obtained illustrate that the proposed algorithm finds promising results.

6. Conclusions and future research directions

In our study, a new two sided U-type assembly line balancing problem with sequence-dependent set-up times model and a new ACO algorithm in order to solve this model with the objectives of minimizing the number of positions and the number of total stations for a given cycle time is presented. Exploring capability of the proposed ACO algorithm is reinforced by using a heuristic priority rule based procedure. An illustrative example is used in order to explain each step of the proposed algorithm. Test problems taken from the literature and newly generated data set are solved to prove the efficiency of the proposed algorithm. Obtained results are compared with the best known U-type assembly line balancing problem without

set-up times. Initial solution using six types of priority rules are improved for the proposed ACO. The experimental results and statistical analysis show that the proposed approach is efficient in solving the TUALBPS. An efficient lower bound will be developed to analyse performance of the proposed algorithm at future research. Moreover, another population based metaheuristics such as new variants of ACO, ABC or PSO algorithms will be proposed for the problem. Also, for the future research multi objective, mixed model or stochastic versions of the problem may be a promising direction.

Acknowledgement

This research was supported by Scientific Research Fund of Erciyes University under the contract no: FBA-2017-7349.

Notations

IS	Iteration size (number of iterations)
$iter$	Iteration index $(1 \leq iter \leq IS)$
CS	Colony size
a	Colony index $(1 \leq a \leq CS)$
n	Number of tasks
i,j	Task index $(1 \leq i,j \leq n)$
CL	A list composed of candidate tasks
$PM_{i,j}$	Precedence matrix which keeps the precedence relations between all tasks
$TM_{i,m}$	Task matrix which keeps the required values of each task
$\tau_{t,i}$	Pheromone matrix saves real numbers which indicate the pheromone trail intensity of the task i stored in the t^{th} task assignment process
$\eta_{a,t}$	Heuristic information matrix saves one of the six different heuristic information which is required to calculate the selection probability (P) of t^{th} task assignment process for the ant a
$S_{a,i,k}$	Solution matrix saves detailed solutions for each task (i) of each ant
$SR_{a,l}$	Solution Result matrix saves objective function values for each ant
NP	Position index
NS	Station index
loc	Assignment locations, (loc = 1,2, 3, 4)
pos	The selected position for assignment, (pos = 1, 2, …, $\text{pos}^{\max}$)
C	Cycle time
t_i	Task time of each task, $i \in \{1,2,\ldots,n\}$
$fs_{i,j}$	Forward set-up time for all $i,j \in \{1,2,\ldots,n\}$
$bs_{i,j}$	Backward set-up time for all $i,j \in \{1,2,\ldots,n\}$
$cs_{i,j}$	Crossover set-up time for all $i,j \in \{1,2,\ldots,n\}$
$FSM_{i,j}$	Forward set-up matrix consists of the setup values between each task, for all i,j where $i,j \in \{1,2,\ldots,n\}$
$BSM_{i,j}$	Backward set-up matrix consists of the setup values between each task, for all i,j where $i,j \in \{1,2,\ldots,n\}$
$CSM_{i,j}$	Crossover set-up matrix consists of the setup values between each task, for all i,j where $i,j \in \{1,2,\ldots,n\}$
$F_{pos,loc}$	The feasibility value of the current assignment operation at position pos and location loc
$RT_{pos,loc}$	Remainder time of the current assignment operation at position pos and location loc
$X_{r,i}$	It is used to save all of the priority rule values, which are determined in the initialization step, for all tasks i in CL
pr_i	It is used to save all of the calculated relative priority rule value of each candidate task
P_i	The selection probability value of task i. It is calculated using the ant's pheromone value and the selected priority rule
SP_l	Cumulative selection probability matrix

References

[1] Boysen N, Fliedner M and Scholl A 2007 A classification of assembly line balancing problems. *Eur. J. Oper. Res.* 183: 674–693

[2] Li M, Tang Q, Zheng Q, Xia X and Floudas C A 2017 A Rules-based heuristic approach for the U-shaped assembly line balancing problem. *Appl. Math. Modell.* https://doi.org/10.1016/j.apm.2016.12.031

[3] Becker C and Scholl A 2006 A survey on problems and methods in generalized assembly line balancing. *Eur. J. Oper. Res.* 168: 694–715

[4] Salveson M E 1955 The assembly line balancing problem. *J. Ind. Eng.* 6(6): 18–25

[5] Urban T L and Chiang W C 2006 An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. *Eur. J. Oper. Res.* 168(3): 771–782

[6] Boysen N, Fliedner M and Scholl A 2008 Assembly line balancing: Which model to use when? *Int. J. Product. Econ.* 111: 509–528

[7] Battaia O and Dolgu A 2013 A taxonomy of line balancing problems and their solution approaches. *Int. J. Product. Econ.* 142: 259–277

[8] Sivasankaran P and Shahabudeen P 2014 Literature review of assembly line balancing problems. *Int. J. Adv. Manuf. Technol.* 73: 1665–1694

[9] Kim Y K, Kim Y, Kim Y J 2000 Two-sided assembly line balancing: a genetic algorithm approach. *Product. Plan. Control* 11: 44–53

[10] Bartholdi J J 1993 Balancing two-sided assembly lines: A case study. *Int. J. Product. Res.* 31: 2447–2461

[11] Lee T O, Kim Y and Kim Y K 2001 Two-sided assembly line balancing to maximize work relatedness and slackness. *Comput. Ind. Eng.* 40: 273–292

[12] Miltenburg J and Wijngaard J 1994 The U-line balancing problem. *Manag. Sci.* 40(10): 1378–1388

[13] Urban T L 1998 Optimal balancing of U-shaped assembly lines. *Manag. Sci.* 44(5): 738–741

[14] Aigbedo H and Monden Y 1997 A parametric procedure for multi-criterion sequence scheduling for just-in-time mixed-model assembly lines. *Int. J. Product. Res.* 35: 2543–2564

[15] Miltenburg J 1998 Balancing U-lines in a multiple U-line facility. *Eur. J. Oper. Res.* 109: 1–23

[16] Yegül M F, Ağpak K and Yavuz M 2010 A new algorithm for U-shaped two-sided assembly line balancing. *Trans. Can. Soc. Mech. Eng.* 34(2): 225–241

[17] Ağpak K, Yegül M F and Gökçen, H 2012 Two-sided U-type assembly line balancing problem. *Int. J. Product. Res.* 50(18): 5035–5047

[18] Delice Y, Aydoğan E K, Özcan and İlkay M S 2017 Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm. *4OR* 15: 37–66

[19] Delice Y, Kızılkaya Aydoğan E and Özcan U 2016 Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach. *Int. J. Product. Res.* 54(11): 3429–3451

[20] Andrés C, Miralles C and Pastor R 2008 Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *Eur. J. Oper. Res.* 187(3): 1212–1223

[21] Scholl A, Boysen N and Fliedner M 2008 The sequence-dependent assembly line balancing problem. *OR Spectr.* 30(3): 579–609

[22] Martino L and Pastor R 2010 Heuristic procedures for solving the general assembly line balancing problem with setups. *Int. J. Product. Res.* 48(6): 1787–1804

[23] Özcan U and Toklu B 2010 Balancing two-sided assembly lines with sequence-dependent setup times. *Int. J. Product. Res.* 48(18): 5363–5383

[24] Nazarian E, Ko J and Wang H 2010 Design of multi-product manufacturing lines with the consideration of product change dependent inter-task times, reduced changeover and machine flexibility. *J. Manuf. Syst.* 29(1): 35–46

[25] Seyed-Alagheband S A, Ghomi S F and Zandieh M 2011 A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. *Int. J. Product. Res.* 49(3): 805–825

[26] Yolmeh A and Kianfar F 2012 An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Comput. Ind. Eng.* 62(4): 936–945

[27] Hamta N, Ghomi S F, Jolai F and Shirazi M A 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. Product. Econ.* 141(1): 99–111

[28] Akpinar Ş, Bayhan G M and Baykasoğlu A 2013 Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Appl. Soft Comput.* 13(1): 574–589

[29] Scholl A, Boysen N and Fliedner M 2013 The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectr.* 35(1): 291–320

[30] Akpinar Ş and Baykasoğlu A 2014 Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model. *J. Manuf. Syst.* 33(1): 177–187

[31] Akpinar Ş and Baykasoğlu A 2014 Modeling and solving mixed-model assembly line balancing problem with setups. Part II: A multiple colony hybrid bees algorithm. *J. Manuf. Syst.* 33(4): 445–461

[32] Esmaeilbeigi R, Naderi B and Charkhgard P 2016 New formulations for the setup assembly line balancing and scheduling problem. *OR Spectr.* 38: 493–518

[33] Şahin M and Kellegöz T 2017 Increasing production rate in U-type assembly lines with sequence-dependent set-up times. *Eng. Optim.* 49(8): 1401–1419

[34] Akpinar Ş, Elmi A and Bektaş T 2017 Combinatorial Benders cuts for assembly line balancing problems with setups. *Eur. J. Oper. Res.* 259(2): 527–537

[35] Gutjahr A L and Nemhauser G L 1964 An algorithm for the line balancing problem. *Manag. Sci.* 11(2): 308–315

[36] Mogale D G, Kumar M, Kumar S K and Tiwari M K 2018 Grain silo location-allocation problem with dwell time for optimization of food grain supply chain network. *Transp. Res. Part E* 111: 40–69

[37] Mogale D G, Kumar S K and Tiware M K 2018 An MINLP model to support the movement and storage decisions of the Indian food grain supply chain. *Control Eng. Pract*. 70: 98–113

[38] Blum C 2005 Beam-ACO - Hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Comput. Oper. Res.* 32(6): 1565–1591

[39] Ding Q, Hu X, Sun L and Wang Y 2012 An improved ant colony optimization and its application to vehicle routing problem with time windows. *Neurocomputing* 98: 101–107

[40] Mogale D G, Dolgui A, Kandhway R, Kumar S K and Tiwari M K 2017 A multi-period inventory transportation model for tactical planning of food grain supply chain. *Comput. Ind. Eng.* 110: 379–394

[41] Dorigo M and Stützle T 2009 Ant colony optimization: Overview and recent advances. Techreport, IRIDIA. Universite Libre de Bruxelles

[42] Colorni A, Dorigo M and Maniezzo V 1991 Distributed optimization by ant colonies. In: *Proceedings of ECAL 91-European Conference on Artificial Life*, Paris, France. Elsevier, Amsterdam, pp. 134–142

[43] Colorni A, Dorigo M and Maniezzo V 1992 An investigation of some properties of an ant algorithm. In: Manner R, Manderick B (Eds.), In: *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92),* Brussels, Belgium. Elsevier, Amsterdam, pp. 509–520

[44] Bautista J and Pereira J 2002 Ant algorithms for assembly line balancing. In: *Lecture Notes in Computer Science* 2463: 65–75

[45] Bautista J and Pereira J 2007 Ant algorithms for a time and space constrained assembly line balancing problem. *Eur. J. Oper. Res.* 177: 2016–2032

[46] McMullen P R and Tarasewich P 2003 Using ant techniques to solve the assembly line balancing problem. *IIE Trans.* 35: 605–617

[47] Sabuncuoglu I, Erel E and Alp A 2009 Ant colony optimization for the single model U-type assembly line balancing problem. *Int. J. Product. Econ.* 120: 287–300

[48] Simaria A S and Vilarinho P M 2009 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Comput. Ind. Eng.* 56, 489–506

[49] Yagmahan B 2011 Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. *Expert Syst. Appl.* 38: 12453–12461

[50] Kucukkoc I and Zhang D Z 2016 Mixed-model parallel two-sided assembly line balancing problem: a flexible agent-based ant colony optimization approach. *Comput. Ind. Eng.* 97: 58–72

[51] Bautista J, Suarez R, Mateo M and Companys R 2000 Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. In: *Proceedings of the IEEE international conference on robotics and automation*. San Francisco, CA, pp. 2404–2409

[52] Helgeson W and Birnie D 1961 Assembly line balancing using the ranked positional weight technique. *J. Ind. Eng.* 12: 394–398

[53] Tonge F 1961 *A Heuristic Program of Assembly Line Balancing*. Englewood Cliffs, NJ: Prentice-Hall

[54] Kilbridge M and Wester L 1961 A heuristic method for assembly line balancing. *J. Ind. Eng.* 12: 292–298

[55] Arcus A L 1963 *An analysis of a computer method of sequencing assembly line operations*. PhD dissertation. University of California, Berkeley

[56] Moodie C L and Young H H 1965 A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *J. Ind. Eng.* 16: 23–29

[57] Brian T F and Patterson J H 1984 An integer programming algorithm with network cuts for solving the assembly line balancing problem. *Manag. Sci.* 30(1): 85–99

[58] Elsayed E A and Boucher T O 1994 *Analysis and Control of Production Systems*. New Jersey: Prentice Hall International Series in Industrial and Systems Engineering

[59] Talbot F B, Patterson J H and Gehrlein W V 1986 A comparative evaluation of heuristic line balancing techniques. *Manag. Sci.* 32(4): 430–454

[60] Scholl A and VoB S 1994 *A note on fast, effective heuristics for simple assembly line balancing*, Working paper, TH Darmstadt

[61] Boctor F F 1995 A Multiple-rule heuristic for assembly line balancing. *J. Oper. Res. Soc.* 46: 62–69