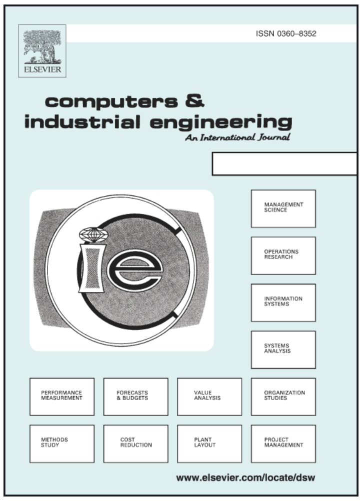


# Type-E Parallel Two-Sided Assembly Line Balancing Problem: Mathematical Model and Ant Colony Optimisation based Approach with Optimised Parameters

Ibrahim Kucukkoc, David Z. Zhang

| | |
|---|---|
| PII: | S0360-8352(14)00466-5 |
| DOI: | http://dx.doi.org/10.1016/j.cie.2014.12.037 |
| Reference: | CAIE 3917 |

To appear in: *Computers & Industrial Engineering*

# Type-E Parallel Two-Sided Assembly Line Balancing Problem: Mathematical Model and Ant Colony Optimisation based Approach with Optimised Parameters

Ibrahim KUCUKKOC [ab*] and David Z. ZHANG [ac]

[a] College of Engineering Mathematics and Physical Sciences, North Park Road, University of Exeter, EX4 4QF, Exeter, England

[b] Department of Industrial Engineering, Balikesir University, Cagis Campus, 10145, Balikesir, Turkey

[c] State Key Laboratory on Mechanical Transmission, Chongqing University, Chongqing 400044, China

I.Kucukkoc@exeter.ac.uk, D.Z.Zhang@exeter.ac.uk

There are many factors which affect the performance of a complex production system. Efficiency of an assembly line is one of the most important of these factors since assembly lines are generally constructed as the last stage of an entire production system. Parallel two-sided assembly line system is a new research domain in academia though these lines have been utilised to produce large sized products such as automobiles, trucks, and buses in industry for many years. Parallel two-sided assembly lines carry practical advantages of both parallel assembly lines and two-sided assembly lines.

The main purpose of this paper is to introduce type-E parallel two-sided assembly line balancing problem for the first time in the literature and to propose a new ant colony optimisation based approach for solving the problem. Different from the existing studies on parallel assembly line balancing problems in the literature, this paper aims to minimise two conflicting objectives, namely cycle time and number of workstations at the same time and proposes a mathematical model for the formal description of the problem. To the best of our knowledge, this is the first study which addresses both conflicting objectives on a parallel two-sided assembly line configuration. The developed ant colony optimisation algorithm is illustrated with an example to explain its procedures. An experimental design is also conducted to calibrate the parameters of the proposed algorithm using response surface methodology. Results obtained from the performed computational study indicate that minimising cycle time as well as number of workstations help increase system efficiency. It is also observed that the proposed algorithm finds promising results for the studied cases of type-E parallel two-sided assembly line balancing problem when the results are compared with those obtained from other three well-known heuristics.

**Keywords:** Parallel two-sided assembly lines; Type-E assembly line balancing; Ant colony optimisation; Response surface methodology; Artificial intelligence.

* Corresponding author: Ibrahim Kucukkoc. Permanent Email: ikucukkoc@balikesir.edu.tr

Tel: +441392 723613 (IK); +441392 723641 (DZZ)

# Type-E Parallel Two-Sided Assembly Line Balancing Problem: Mathematical Model and Ant Colony Optimisation based Approach with Optimised Parameters

## Abstract

There are many factors which affect the performance of a complex production system. Efficiency of an assembly line is one of the most important of these factors since assembly lines are generally constructed as the last stage of an entire production system. Parallel two-sided assembly line system is a new research domain in academia though these lines have been utilised to produce large sized products such as automobiles, trucks, and buses in industry for many years. Parallel two-sided assembly lines carry practical advantages of both parallel assembly lines and two-sided assembly lines.

The main purpose of this paper is to introduce *type-E parallel two-sided assembly line balancing problem* for the first time in the literature and to propose a new ant colony optimisation based approach for solving the problem. Different from the existing studies on parallel assembly line balancing problems in the literature, this paper aims to minimise two conflicting objectives, namely *cycle time* and *number of workstations* at the same time and proposes a mathematical model for the formal description of the problem. To the best of our knowledge, this is the first study which addresses both conflicting objectives on a parallel two-sided assembly line configuration. The developed ant colony optimisation algorithm is illustrated with an example to explain its procedures. An experimental design is also conducted to calibrate the parameters of the proposed algorithm using response surface methodology. Results obtained from the performed computational study indicate that minimising cycle time as well as number of workstations help increase system efficiency. It is also observed that the proposed algorithm finds promising results for the studied cases of *type-E parallel two-sided assembly line balancing problem* when the results are compared with those obtained from other three well-known heuristics.

**Keywords:** Parallel two-sided assembly lines; Type-E assembly line balancing; Ant colony optimisation; Response surface methodology; Artificial intelligence.

## 1. Introduction

Assembly lines are widely used flow-oriented production systems designed to produce high-quality and low-cost standardised homogeneous products, and have been a matter of concern of researchers for decades. An assembly line consists of serially linked workstations (with a conveyor belt or material handling system), in which a group of tasks is performed according to given precedence relationships within a limited duration (cycle time) (Avikal *et al.*, 2013; Kara *et al.*, 2011; Scholl and Boysen, 2009). Assembly line balancing problem is to assign tasks to an ordered sequence of workstations optimally by satisfying specific constraints (*i.e.* capacity constraints, assignment constraints, precedence constraints, *etc.*) (Kucukkoc *et al.*, 2013; Tuncel and Topaloglu, 2013). Each task must be assigned to exactly one workstation. The sum of processing times of all tasks assigned to a workstation constitutes its workload time and cannot exceed the cycle time designated for this workstation (Khorasanian *et al.*, 2013).

The studies related to assembly line balancing problems can be classified into two general groups according to the implementation of the lines: 'traditional assembly lines' and 'parallel assembly lines'. While traditional lines do not address line parallelisation; in parallel assembly lines, two or more lines are located in parallel to each other to maximise the sharing of resources and tools. Although the literature on traditional lines is rather extensive, the number of studies on Parallel Assembly Line Balancing Problem (PALBP) is quite limited. Table 1 summarises the main contributions regarding parallel assembly line balancing problems and lists out the proposed approaches till now.

The parallel line configuration idea was first addressed by Suer and Dagli (1994). They proposed a heuristic procedure which aims at determining the number of lines and workstations by considering assigning different models of a product to the lines. However, the precedence constraints were not considered and it was assumed that the entire job could be divided into any number of operations. Afterwards, Suer (1998) proposed alternative line configuration strategies for a single product.

Table 1. Summary of the literature on parallel assembly line balancing problems, adapted from Kucukkoc and Zhang (2014c).

| **Research** | **Method / approach** | **PM** | | | | **Additional aims/features** |
|---|---|---|---|---|---|---|
| | | **K** | **C** | **O** | | |
| Suer and Dagli (1994) | Heuristic procedure | ● | | | | Dynamic number of lines |
| Suer (1998) | 3-phase heuristic with IP and MILP model | ● | | | | Dynamic number of lines |
| Gökçen *et al.* (2006) | Heuristic procedures and a mathematical programming model | ● | | | | |
| Benzer *et al.* (2007) | A network model | ● | | | | |
| Lusa (2008) | Survey | | | | | |
| Baykasoglu *et al.* (2009) | Ant colony optimisation | ● | | | | |
| Cercioglu *et al.* (2009) | Simulated annealing based approach | ● | | | | |
| Ozcan *et al.* (2009) | Tabu search algorithm | ● | | | | Workload balance between workstations |
| Scholl and Boysen (2009) | Binary linear programme and SALOME based exact solution procedure | ● | | ● | | Product-line assignment |
| Kara *et al.* (2010) | Two goal programming approaches | ● | ● | | | Three conflicting goals, task loads of workstations |
| Ozcan *et al.* (2010a) | Simulated annealing algorithm | ● | | | | Mixed-models and model sequencing, workload variance between workstations |
| Ozcan *et al.* (2010b) | Tabu search algorithm | ● | | | | Parallel two-sided lines |
| Kucukkoc and Zhang (2014c) | Framework of a possible solution approach | ● | | | | Line length, mixed-model parallel two-sided lines, model sequencing |
| Kucukkoc and Zhang (2014b) | Agent based ant colony optimisation algorithm | ● | | | | Line length, mixed-model parallel two-sided lines, model sequencing |

PM: Performance measure, K: Number of stations, C: Cycle time, O: Number of operators, IP: Integer programming, MILP: Mixed-integer linear programming.

However, the real PALBP, balancing of two or more assembly lines with a common set of resources, was introduced by Gökçen *et al*. (2006). Gökçen *et al*. (2006) formulated the

PALBP mathematically and proposed two heuristic approaches. Development of other heuristic/meta-heuristic approaches followed Gökçen *et al*. (2006) and Benzer *et al*. (2007) proposed a new shortest path approach based model for PALBP and illustrated the performance of the model on a numerical example. Baykasoglu *et al*. (2009) proposed a novel Ant Colony Optimisation (ACO) based algorithm for PALBP and compared their test results with three other existing approaches from the literature. Cercioglu *et al*. (2009) proposed a simulated annealing approach to solve PALBP and compared their results with the results of existing heuristic algorithm proposed by Gökçen *et al*. (2006). Ozcan *et al*. (2009) developed first multi-objective tabu search algorithm for PALBP and tested the performance of the algorithm on a set of well-known problems in the literature. Scholl and Boysen (2009) modelled the PALBP mathematically and proposed an exact solution procedure. Kara *et al*. (2010) suggested a fuzzy goal programming model that could be used for balancing parallel assembly lines. Ozcan *et al*. (2010a) addressed parallel mixed-model assembly line balancing and sequencing problem with a simulated annealing approach with the aim of maximising the line efficiency by considering workload smoothness among workstations. Ozbakir *et al*. (2011) developed a novel multiple-colony ant algorithm for balancing bi-objective parallel assembly lines. This was one of the first attempts to solve the problem with swarm intelligence based meta-heuristics. Please refer to Lusa (2008) and Zhang and Kucukkoc (2013) for a more detailed survey on multiple and parallel assembly line balancing problems.

Assembly lines can alternatively be classified as one-sided assembly lines and two-sided assembly lines. While only one side of the line is used in a one sided assembly line, both left and right sides are used parallel in two-sided assembly lines. Two-sided assembly lines, introduced by Bartholdi (1993) for the first time, are usually designed to produce highvolume large-sized products such as trucks and buses. To solve the two-sided assembly line balancing problem some exact solution approaches were developed by Wu *et al.* (2008), Hu *et al.* (2010); and some heuristic/meta-heuristic approaches were proposed by Kim *et al.* (2000), Lee *et al.* (2001), Hu *et al.* (2008), Kim *et al.* (2009), Ozcan and Toklu (2009, 2010), Yegul *et al.* (2010), Ozcan (2010), Ozbakir and Tapkan (2010, 2011), Taha *et al.* (2011), Chutima and Chimklai (2012), Rabbani *et al.* (2012), Purnomo *et al.* (2013), Khorasanian *et al*. (2013), and Tuncel and Aydin (2014). Among proposed meta-heuristics, studies belong to Baykasoglu and Dereli (2008), and Simaria and Vilarinho (2009) represented implementation of different ACO algorithms to balance two-sided lines with success.

Although the combination of the aforementioned types of production lines (parallel lines and two-sided lines) are frequently used in producing large sized items in industry, Parallel Two-sided Assembly Line Balancing Problem (PTALBP) was introduced by Ozcan *et al.* (2010b) very recently and there is only one published research concerning this problem so far. The reason for this situation could be the complexity of the PTALBP, as there is more than one line to be balanced and different conditions (*i.e.* precedence relationships, cycle times, task processing times, *etc.*) exist on each of the lines. Also, disregarding (or ignoring) the advantages of multi-line stations, which can be established between two adjacent lines, could contribute to the lack of studies on PTALBP. However, Ozcan *et al.* (2010b) described the concept of parallel two-sided assembly lines and showed the advantage of utilising multi-line stations by comparing their results obtained through two scenarios: *(i)* balancing the parallel two-sided lines together, where multi-line stations are allowed, and *(ii)* balancing the parallel two-sided lines individually, where multi-line stations are not allowed. It was clear that balancing lines together by allowing utilisation of multi-line stations yields better performance measures.

In terms of the sought performance measure, line balancing problems could be classified in four groups:

- Type-1: Minimises number of workstations, given cycle time.
- Type-2: Minimises cycle time, given number of workstations.
- Type-E: Minimises both number of workstations and cycle time.
- Type-F: Searches a feasible solution for a given number of workstations and cycle time.

As summarised in Table 1, the majority of the studies on PALBP only minimises the number of workstations (this problem is referred to as type-1) as an ultimate goal. The only study addressing multi-objective goals, namely number of workstations and cycle time, belongs to Kara *et al.* (2010) in this domain. Specifically, in the PTALBP literature, the unique study that belongs to Ozcan *et al.* (2010b) ignores minimisation of cycle time and deals with number of workstations (type-1).

Based on this motivation, this paper addresses type-E PTALBP (referred to as PTALBP-E hereafter) to fill in the gap in the literature as pointed out above. The main aim is to describe PTALBP-E and to propose a new possible solution approach, which is an ACO algorithm whose parameters are optimised through a well-known design of experiment technique - Response Surface Methodology (RSM). ACO algorithm is selected because of its several successful implementations in solving hard combinatorial optimisation problems, and various assembly line balancing problems in particular; such as McMullen and Tarasewich (2003, 2006), Vilarinho and Simaria (2006), Baykasoglu (2008), Blum (2008) Baykasoglu *et al.* (2009), Simaria and Vilarinho (2009), Sabuncuoglu *et al.* (2009), Yagmahan (2011), Ozbakir *et al.* (2011), Fattahi *et al.* (2011) and Akpinar *et al.* (2013).

This paper contributes to the knowledge by not only introducing the type-E parallel two-sided assembly line balancing problem for the first time in the literature, but also solving this problem using a powerful ACO based approach enhanced with three commonly used heuristics in the line balancing domain. By this way, two conflicting objectives, namely minimisation of cycle times and minimisation of number of workstations, are handled for a parallel two-sided line system for the first time in the literature. This is important because a common cycle time needs to be established for each different combination (or pair) of cycle times belonging to each of the parallel lines. In addition, a common problem faced in meta-heuristic implementations, calibration of the used parameters, is overcome by determining the ACO parameters using a well-known design of experiment technique, RSM. This is another significant contribution of this research as RSM is used for the purpose of calibrating ACO parameters for the first time in the entire line balancing literature.

The rest of the paper is organised as follows. Section 2 presents the main characteristics of the parallel two-sided assembly lines alongside the assumptions considered. The proposed ACO based approach is described in Section 3 comprehensively, and illustrated with an example in Section 4. Section 5 optimises the parameters of the developed ACO algorithm first, followed by the results of the computational study. Finally, Section 6 concludes with the findings of the research and the future research directions.

## 2. Parallel Two-Sided Assembly Lines

### 2.1. Main characteristics

Parallel two-sided assembly lines are mainly used to produce one or more similar product models that have similar production processes in a set of two-sided assembly lines constructed in parallel to each other. Typical illustration of a parallel two-sided line system is depicted in Figure 1.

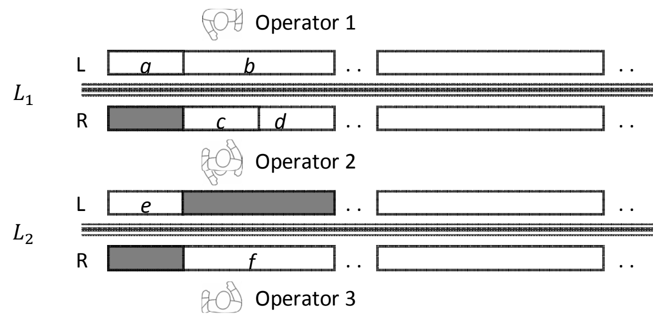


Figure 1. Configuration of parallel two-sided assembly lines, adapted from (Ozcan *et al.*, 2010b).

The type-E parallel two-sided assembly line balancing problem is to balance two or more two-sided assembly lines, which are constructed in parallel, to optimise the system efficiency by minimising two conflicting objectives: cycle time and number of workstations. Tasks are allocated to workstations by considering precedence relationships (which may be caused by technological priorities or organisational structures) and capacity constraints. Different product models are produced on different two-sided assembly lines, $L_h$ ($h = 1, \dots, H$), and each product model has its own set of tasks, $t_{hi}$ ($i = 1, \dots, T_h$). These tasks are performed according to predefined precedence relationships among tasks. $P_h$ represents the set of task pairs that have precedence relationships in between each other on line $L_h$. Each task, which is performed on line $L_h$, needs a certain amount of time symbolised with $pt_{hi}$; and each line consists of a series of workstations, $W_{hkj}$ ($k = 1, \dots, K_h$, and $j = 0, 1$) where $j$ represents the operation side of the line (Ozcan *et al.*, 2010b).

An advantage of this line system is that each line may have a different cycle time ($C_h$), which increases flexibility. In that case, a common cycle time should be used to assign tasks in each cycle. Gökçen *et al.* (2006) used least common multiple ($LCM$) based approach for different cycle time situation of two parallel lines (Ozcan *et al.*, 2010b). In this approach (Gökçen *et al.*, 2006):

- least common multiple of the cycle times is found,
- integers $ld_1$ and $ld_2$ are calculated via dividing the $LCM$ value by the cycle times of $L_1$ and $L_2$ ($C_1$ and $C_2$) respectively,
- task times of the product models produced on the $L_1$ and $L_2$ are multiplied by $ld_1$ and $ld_2$, separately,
- $LCM$ is determined as the common cycle time ($C$) of the lines and the lines are balanced together.

Another advantage of the parallel two-sided line system is the flexibility of implementing multi-line stations. Stations can be utilised either in only one or in two adjacent two-sided lines. As can be seen in Figure 1, three operators are needed to perform tasks $a-f$, and Operator-2 first completes task *e* at the left side station of $L_2$, and then tasks *c* and *d* at the right side station of $L_1$. Please note that the shades in the figure symbolise idle times.

More attention is needed to acquire a feasible solution when balancing two-sided assembly lines. The precedence relationships among tasks should be considered carefully since tasks, which have precedence relationships with each other and are performed on different sides of each line, must be assigned considering finishing time of previously assigned tasks. Let us consider $P_1$ as set of precedence relationships of tasks on Line I. If $(a,b) \in P_1$ and $(a,c) \in P_1$, then tasks $b$ and $c$ can be initialised after completion of task $a$, which may be performed at the other side of the line. This phenomenon is called *interference* in the literature and the violation of this rule yields infeasible balancing solutions.

The following sub-section provides the mathematical model of the PTALBP-E based on the formulation proposed by Gökçen *et al.* (2006) for PALBP (without two-sided configuration).

## 2.2. Mathematical model

The notation used in the mathematical model proposed in this study can be summarised as follows:

$H$: The number of lines subject to balancing,

$L_h$, $(h = 1, \dots, H)$: The $h^{th}$ line. As there is only one product model on each of the lines, this index will be used for the representation of product model assembled on each line as well.

$T_h$: The total number of tasks performed on the line $L_h$,

$t_{hi}$, $(i = 1, \dots, T_h)$: The $i^{th}$ task on the line $L_h$,

$$j = \begin{cases} 0 & \text{indicates left side of relevant line} \\ 1 & \text{indicates right side of relevant line'} \end{cases}$$

$K_h$: The total number of workstations utilised across the line $L_h$,

$W_{hkj}, (k = 1, \dots, K_h;\ j = 0, 1)$: The $k^{th}$ workstation on line $L_h$,

$pt_{hi}$: The processing time of task $t_{hi}$ on the line $L_h$,

$C$: The common cycle time for all lines,

$P_h$: Set of precedence relationships in precedence diagram of line $L_h$,

$st_{hi}$ : Starting time of task $t_{hi}$ on line $L_h$,

$q_{hjk}$: Queue number that station $W_{hkj}$ is utilised on,

$$S_k = \begin{cases} 0 & \text{if station } W_{hkj} \text{ is utilised on left side of the first line} \\ 1 & \mathit{otherwise} \end{cases},$$

$$z_{hjk} = \begin{cases} 1 & \text{if any task is assigned to station } W_{hkj} \\ 0 & \mathit{otherwise} \end{cases},$$

$$U_{hjk} = \begin{cases} 1 & \text{if station } k \text{ is utilised on side } j \text{ of line } L_h \\ 0 & \mathit{otherwise} \end{cases},$$

$\delta$ : Variable, $(\delta = h + 1, \dots, H)$,

$\gamma$ : Variable, $(\gamma \in \{0, 1\})$,

$$c = \begin{cases} 1 & \text{if } j = 1 \text{ and } \gamma = 1 \\ 0 & otherwise \end{cases},$$

$$\mu = \begin{cases} 1 & \text{if } (\delta - h) = 1 \; and \; \gamma = 1 \\ 0 & otherwise \end{cases},$$

$$R_{huv} = \begin{cases} 1 & \text{if tasks } u \text{ and } v \text{ are assigned to the same workstation on line } L_h \\ 0 & otherwise \end{cases},$$

$$X_{hijk} = \begin{cases} 1 & \text{if task } t_{hi} \text{ is assigned to workstation } W_{hkj}, \text{on side } j \text{ of line } L_h \\ 0 & otherwise \end{cases}.$$

### 2.2.1. Objective function

In type-E assembly line balancing problems, both cycle time and number of workstations are minimised at the same time, as explained in Section 1. Therefore, the objective function given in Equation (1) aims to minimise the product of cycle time and total number of workstations needed. Please note that the cycle time for all of the lines will be the same with this model.

$$Min\, Z = C \cdot \sum_{h=1}^{H} \sum_{j \in \{0,1\}} \sum_{k=1}^{K_h} z_{hjk} \,. \tag{1}$$

### 2.2.2. Constraints

*Assignment Constraint:*

As in majority of the researches on assembly line balancing problems, this research also assumes that tasks cannot split into two or more workstations. Therefore, a task can only be assigned to exactly one workstation. For this aim, constraint given in Equation (2) ensures that each task is assigned to a workstation exactly once. In other words, sum of workstations in which a task is assigned must equal to '1' and this is applied for all tasks on all lines.

$$\sum_{j \in \{0,1\}} \sum_{k=1}^{K_h} X_{hijk} = 1, \qquad \forall i = 1, \dots, T_h; \quad \forall\, h = 1, \dots, H. \tag{2}$$

*Capacity Constraint:*

Workload of a workstation is constituted by the sum of processing times of tasks assigned to that workstation and it cannot exceed the designated cycle time ($C$), if that workstation is utilised ($z_{hjk} = 1$). Constraint (3) assures that total workload of a workstation cannot exceed the cycle time. Different from traditional assembly line balancing problems, this constraint must be applied carefully as multi-line stations are allowed in parallel assembly line systems. In a multi-line station, processing times of tasks assigned to that workstation from the adjacent line must also be considered and the second term of the constraint corresponds to this issue. $S_k$ is a controller and becomes '0' if the considered workstation is on the left side of the first line as it is not possible to utilise a multi-line station at this position. Another position where it is not possible to construct a multi-line station is the right side of the last line and utilisation of multi-line station at this position is restricted by taking $h$ between 1 and $H-1$, ($h = 1, \dots, H-1$).

$$\sum_{i=1}^{T_h}(pt_{hi} + st_{hi}) \cdot X_{hijk} + S_k \cdot \left(\sum_{i=1}^{T_h}(pt_{(h+1)i} + st_{(h+1)i}) \cdot X_{(h+1)i(j-1)k}\right) \leq C \cdot z_{hjk}, \qquad \forall\, k = 1, \dots, K_h; \quad \forall h = 1, \dots, H-1; \quad \forall\, j \in \{0, 1\}. \tag{3}$$

*Precedence Relationships Constraint:*

Precedence relationships constraint is essential for all types of assembly line balancing problems and is satisfied with constraint (4) for this problem. If we consider $(u, v) \in P_h$, which means that task $u$ is a predecessor of task $v$, there are two possibilities to have a feasible assignment solutions: *(i)* task $u$ is assigned to an earlier queue than task $v$ is assigned (the first term of the constraint is active), *(ii)* task $u$ is assigned to the same queue with task $v$ but task $u$ is started and completed before task $v$ is started (the second term of the constraint is active). To remind, $R_{huv}$ gets '1' if tasks $u$ and $v$ are assigned to the same queue to guarantee task $u$ is completed before task $v$ is started.

$$\sum_{j\in\{0,1\}}\sum_{k=1}^{K_h} q_{hjk}\cdot\left(X_{hujk}-X_{hvjk}\right)+R_{huv}\cdot\left(st_{hu}+pt_{hu}-st_{hv}\right)\leq 0,\qquad \forall h = 1,\dots,H;\quad \forall(u,v)\in P_h. \tag{4}$$

*Multi-line Station Utilisation Constraints:*

Constraints (5) and (6) define the utilisation of multi-line stations. In constraint (5), it is guaranteed that the total number of tasks assigned to a workstation is lower than or equal to the total number of tasks on that line if the workstation is utilised.

$$\sum_{i=1}^{T_h} X_{hijk}-T_h\cdot U_{hjk}\leq 0,\qquad \forall k=1,\dots,K_h;\quad \forall h=1,\dots,H;\quad \forall j\in\{0,1\}. \tag{5}$$

Constraint (6) guarantees that an operator working at workstation $W_{hkj}$ can perform additional task(s) from only one adjacent line unless workstation $W_{hkj}$ is utilised on the left side of the first line or on the right side of the last line. To remind, utilisation of multi-line stations on the left side of the first line or on the right side of the last line is not possible due to the nature of the considered assembly line system. Apparently, the first term becomes active and controls the utilisation of multi-line stations when $j=0$ while the second term becomes active when $j=1$. As indicated above, $c$ gets '1' when $j=1$ and $\gamma=1$ while $\mu$ gets '1' when $(\delta-h)=1$ and $\gamma=1$. Both of the variables get '0' in all other situations. To give an example for construction of the multi-line stations, an operator located on the right side of the first line $(h=1,\ j=1)$ can perform some additional tasks from only the left side of the second line $(h=2,\ j=0)$ as well as their main job. As it is not possible to have a direct communication with tasks assigned to the left side of the first line $(h=1, j=0)$ or the right side of the second line $(h = 2, j = 1)$, the above operator cannot perform any job from these two sides.

$$|j-1| \cdot \left(U_{h\gamma k} + U_{(\delta-\mu)jk}\right) + j \cdot \left(U_{h(j-1+c)k} + U_{\delta jk}\right) = 1, \qquad \forall k = 1, \dots, K_h; \quad \forall h = 1, \dots, H; \quad \forall j \in \{0, 1\}; \quad \forall \delta = h+1, \dots, H; \quad \forall \gamma \in \{0, 1\}. \tag{6}$$

### 2.3. Assumptions

The assumptions considered in the study are as follows:

- Only one product model is assembled on each of the lines, so total number of lines equal to total number of product models assembled.
- Each product model has its own precedence relationships diagram.
- Tasks can only be assigned to a predetermined side (Left-L or Right-R) or Either (E).
- The precedence relationships and task times of each product model are known,
- The operators have no preference about the tasks and workstations,
- Walking times of the operators are ignored.

## 3. Proposed ACO Approach for PTALBP-E

ACO is an efficient swarm optimisation technique originated from the foraging behaviour of ants in nature. Its solution approach is motivated by the biological process of finding the shortest path between the nest and the food. The solution of an optimisation problem is a sequence of visited edges (called path) which represent the specific parameters of a solution (Chen *et al.*, 2013).

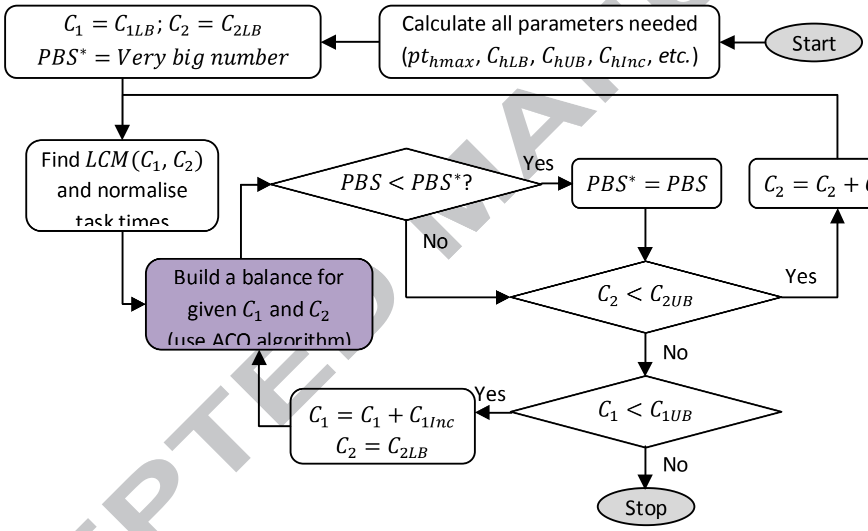
ACO algorithm, which is an improved version of *ant system* proposed by Dorigo *et al*. (1996), is inspired by observation of real ant colonies in the nature. Thanks to their foraging behaviour, ants have capability of finding the shortest path linking the nest and food source.

A substance, called pheromone, is deposited on the ground while they are walking and a pheromone trail is formed by this way. Ants smell pheromone to choose their way in probability and paths involving strong pheromone levels have more chance to be selected by ants (Dorigo *et al.*, 1999). When a set of possible paths is given to the colony, each ant choses one path. Ants picking the shortest path will return faster and there will be more pheromone on the shortest path, influencing later ants to follow this path. By time, the path that has high level pheromone will be most often selected and considered as the shortest route (Leung *et al.*, 2010).

We applied ACO to the PTALBP-E since ACO is a nature based optimization technique whose performance and efficiency has been proven on variants of many combinatorial optimisation problems; such as *traveling salesman problem* (see for example Dorigo and Gambardella (1997), Cheng and Mao (2007), Chen and Chien (2011), Mavrovouniotis and Yang (2013) and Escario *et al.* (2015)), *vehicle routing problem* (see for example Yu *et al.* (2009), Balseiro *et al.* (2011), Yu and Yang (2011), Venkata Narasimha *et al.* (2013) and Reed *et al.* (2014)), *scheduling problem* (see for example Shyu *et al.* (2004), Yagmahan and Yenisey (2008), Deng and Lin (2011) and Tavares Neto and Godinho Filho (2013)), and *assembly line balancing problem* (see for example Sabuncuoglu *et al.* (2009), Simaria and Vilarinho (2009), Ozbakir *et al.* (2011), Yagmahan (2011), Rabbani *et al.* (2012), Akpinar *et al.* (2013) and Kucukkoc and Zhang (2014b)) with numerous successful applications. In ACO, search space of the problem is scanned more effectively with multiple starting points and using both of the exploration and exploitation techniques in comparison with other neighbourhood search based techniques. As ACO mimics the natural behaviour of ants, it also has more capacity to find near optimal solutions by avoiding getting stuck in local minima. The characteristics of the implemented ACO approach within the scope of this study will be explained below.

## 3.1. Outline

The outline of the proposed ACO based algorithm is exhibited in Figure 2. As can be seen from the figure, the algorithm starts by calculating all parameters needed; including maximum task processing times ($pt_{hmax}$), lower and upper bounds of the cycle times ($C_{hLB}$ and $C_{hUB}$), and increments ($C_{hInc}$) of the cycle times for each of the lines. Lower bound values are assigned to cycle times of $L_1$ and $L_2$ ($C_1 = C_{1LB}$, $C_2 = C_{2LB}$) and best optimal solution value is initialised ($PBS^* = \text{Very big number}$). Common cycle time is calculated and task times are normalised as explained in Section 2.1.



$C_1 = C_{1LB}$; $C_2 = C_{2LB}$
$PBS^* = Very\ big\ number$
Calculate all parameters needed ($pt_{hmax}$, $C_{hLB}$, $C_{hUB}$, $C_{hInc}$, etc.)
Start
Find $LCM(C_1, C_2)$ and normalise task times
$PBS < PBS^*$?
Yes
No
$PBS^* = PBS$
$C_2 = C_2 + C_{2Inc}$
Build a balance for given $C_1$ and $C_2$ (use ACO algorithm)
$C_2 < C_{2UB}$
Yes
No
$C_1 = C_1 + C_{1Inc}$
$C_2 = C_{2LB}$
Yes
$C_1 < C_{1UB}$
No
Stop

Figure 2. Flowchart of the proposed solution approach for the PTALBP-E

The parallel two-sided assembly line balancing problem is solved using the ACO approach (see Section 3.2) for the determined parameter values and the global best solution is updated if a better solution is found from the current ACO operation. While $C_1$ remains the same, $C$ is increased by $C_{2Inc}$ and a new balancing solution is built for the new cycle time pair. When $C_2$ reaches its upper bound, $C_1$ is increased by $C_{1Inc}$ and $C_2$ is set to its lower bound ($C_{2LB}$). These cycles continues until $C_1$ and $C_2$ reach their upper bounds. By this way, a new balancing solution is built for each combination of $C_1$ and $C_2$ between the associated lower and upper bounds, and finally the solution which has the maximum system efficiency value is designated as the solution of the problem.

## 3.2. Ant colony optimisation

ACO is run for every pair of cycle time combinations on the lines. Procedures of the ACO algorithm and the process of building a balancing solution are illustrated in Figure 3. As can be seen from this figure, for each side of each line, tasks with no predecessor and satisfying capacity constraints are selected by ants from relevant list and allocated to the workstations one by one; followed by those tasks whose predecessors have been processed and allocated to the workstations, and so on. To prevent infeasible assignments, a timeline is recorded for each workstation, where $st(k)$ and $st(\underline{k})$ represent station time of the current workstation and its mated (opposite) workstation, respectively.

A new ant is released until the colony size is complete and each ant builds a balancing solution by selecting and assigning tasks to the workstations. To increase diversity, balancing starts from a randomly selected line and operation side. Available tasks are determined for the current position (line and side) of ant and a task is selected by the ant and allocated to the current position using pheromone trail and heuristic information. Heuristic information is provided by one of the three well-known line balancing heuristics, named *COMSOAL* (Arcus, 1966), *Ranked Positional Weight Method* (Helgeson and Birnie, 1961), *and Shortest Processing Time* (Baykasoglu, 2006). Each ant is assigned one of these heuristics at random when it is created; and the assigned heuristic is used by this ant until it completes a whole tour. The selection probability of a task by an ant is calculated using the following equation (Kucukkoc and Zhang, 2014a):

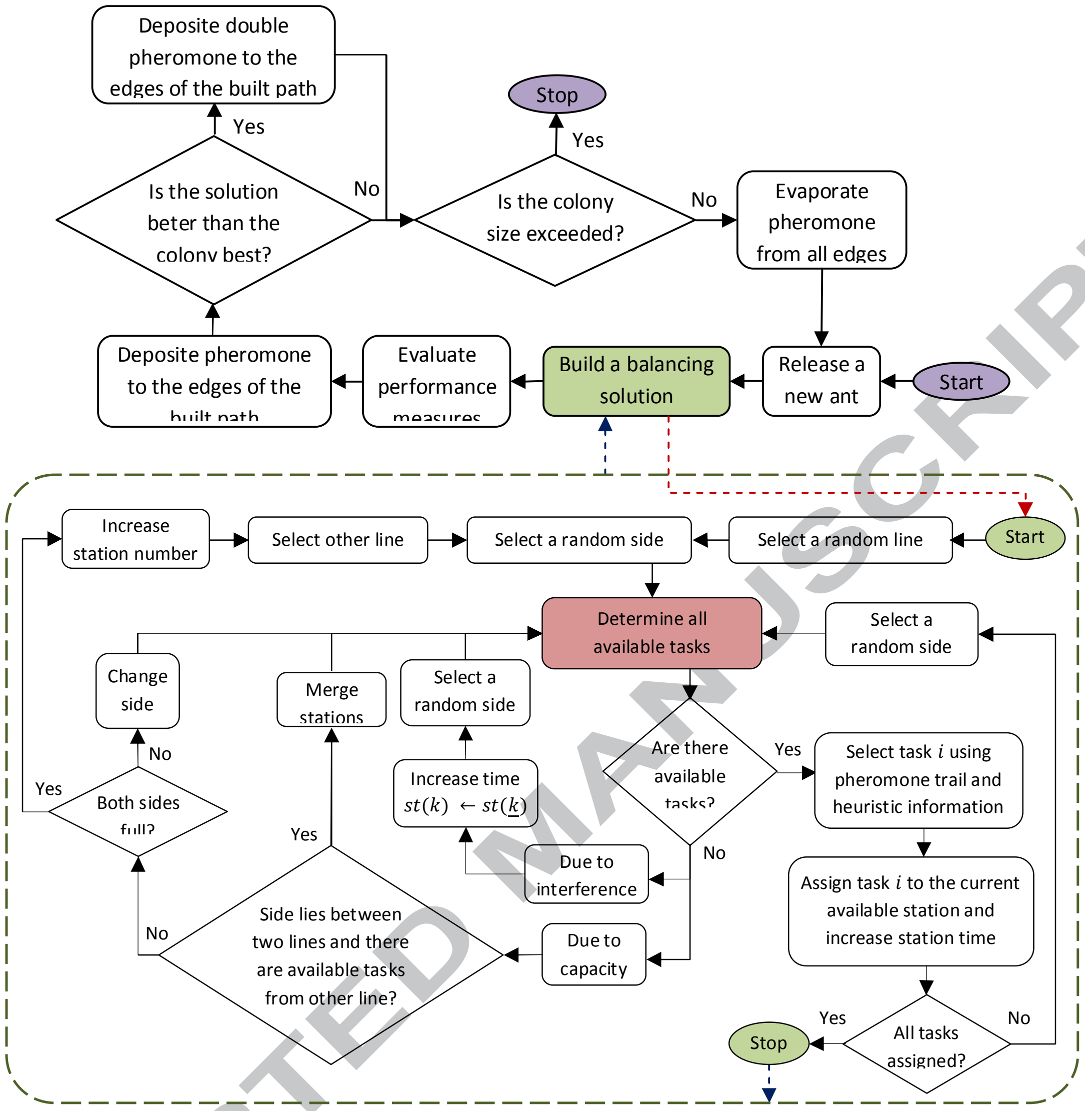

Deposite double pheromone to the edges of the built path
Yes
Is the solution beter than the colony best?
No
Stop
Yes
Is the colony size exceeded?
No
Evaporate pheromone from all edges
Deposite pheromone to the edges of the built path
Evaluate performance measures
Build a balancing solution
Release a new ant
Start
Increase station number
Select other line
Select a random side
Select a random line
Start
Determine all available tasks
Select a random side
Change side
Merge stations
Select a random side
No
Yes
Both sides full?
Increase time $st(k) \leftarrow st(k)$
Are there available tasks?
Yes
Select task $i$ using pheromone trail and heuristic information
No
Due to interference
Yes
No
Side lies between two lines and there are available tasks from other line?
Due to capacity
Assign task $i$ to the current available station and increase station time
Yes
No
Stop
All tasks assigned?

Figure 3. Procedures of the ACO algorithm and building a balancing solution process

$$p_{ik} = \frac{[\tau_{ik}]^{\alpha} \cdot [\eta_i]^{\beta}}{\sum_{y \epsilon Cand_i} [\tau_{iy}]^{\alpha} \cdot [\eta_i]^{\beta}}, \tag{7}$$

where $i$, $k$, and $Cand_i$ indicate task, current workstation, and list of candidate tasks when task $i$ is selected, respectively. $\tau_{ik}$ and $\eta_i$ are the amount of virtual pheromone between task – workstation, and the heuristic information of task $i$ that comes from the randomly selected heuristic. $\alpha$ and $\beta$ are parameters which control the effect of $\tau_{ik}$ and $\eta_i$. This probability is calculated by each ant every time when a new task will be selected, and obviously tasks which have higher probability will have more chance to be selected.

There may not be any available task when: *(i)* assigned tasks to the other side of the line restrict assignment of unassigned tasks because of the precedence relationships, or *(ii)* the remaining capacity of the current workstation is not enough to perform tasks. In the former situation, the station time of the current workstation is forwarded to the station time of its mated station ($st(k) \leftarrow st(\underline{k})$) and candidate tasks for new side are considered. In the latter situation, operation side is changed if both sides are not full. If both sides are full, ant moves forward to the other line and starts assigning tasks to a randomly selected side. As a practical advantage of parallel lines, workstations can be merged to build a multi-line station. During the task allocation process, if the current side of a line lies between two lines and there is no available task to be assigned from the current line but from the adjacent line, the multi-line station is utilised so that some tasks can be performed from the other line. This cycle continues until all tasks are assigned.

When an ant in the colony completes its tour, performance measure of the obtained solution is evaluated and an amount of pheromone is released to the edges (between task and workstation) of the built path based on the quality of the solution. Also, double amount of pheromone is released if the solution is better than the best among all solutions in the colony so far. The pheromone update rule is given in Equation (8) (Kucukkoc and Zhang, 2014a):

$$\tau_{ik} \leftarrow (1-\rho) \cdot \tau_{ik} + \Delta\tau_{ik} \, , \qquad (8)$$

where $\rho$ and $\tau_{ik}$ represent the evaporation rate and the amount of virtual pheromone between task-workstation, respectively; $\Delta\tau_{ik} = 100/Performance\ measure$.

Figure 4 gives an illustration of the procedure used in this study to determine available tasks.

To determine whether a task is available or not, assignment status of all of its predecessors and the remaining capacity are checked for the current assignment position. A task is designated as available if all of its predecessor tasks are completed and the remaining capacity of the current position (workstation) is large enough to perform this task.

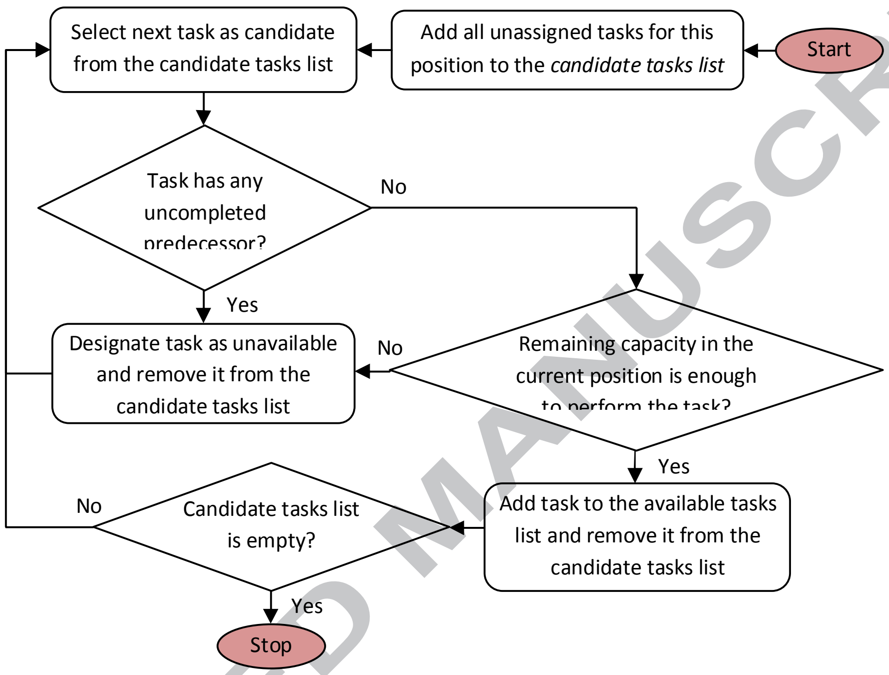

Start
Add all unassigned tasks for this position to the *candidate tasks list*
Select next task as candidate from the candidate tasks list
Task has any uncompleted predecessor?
No
Yes
Designate task as unavailable and remove it from the candidate tasks list
No
Remaining capacity in the current position is enough to perform the task?
Yes
Add task to the available tasks list and remove it from the candidate tasks list
Candidate tasks list is empty?
No
Yes
Stop

Figure 4. The procedure of determining available tasks

## 3.3. Performance measure

Line efficiency is a well-known term which is commonly used as a measure of the obtained solution's quality regardless of the tackled line configuration and the problem type. Therefore, the proximity of a line system's efficiency to '1' could be considered as an indicator whether this system is well balanced or not. If the efficiency equals to '1', this means that there is no idle time on the line. However, this is hardly possible in such systems due to unsmooth task times and different cycle times across the lines.

As this is the first study which deals with a type-E problem in any parallel assembly line configuration, a new equation (see Equation (9)) is formed to calculate the *system efficiency (SE)* of an obtained solution. This equation simply calculates the efficiency of the obtained line balancing solution by dividing *total time spent to perform assigned tasks* to *total time devoted*.

$$SE = \frac{\sum_{h=1}^{H} \frac{C}{C_h} \cdot \left(\sum_{i=1}^{T_h} pt_{hi}\right)}{C \cdot \sum_{h=1}^{H} K_h} \cdot 100. \tag{9}$$

## 4. Illustrative Example

To explain the running mechanism of the proposed algorithm, a numerical example is given in this section. Two well-known test problems, P16 (Lee *et al.*, 2001) for $L_1$ and P24 (Kim *et al.*, 2000) for $L_2$, are taken from the literature and used as input data. Preferred operation sides, processing times and precedence relationships of tasks are presented in Table 2 (where L, R and E symbolise left, right and either sides; respectively).

Table 2. Input data for the numerical example

| $L_1$ (P16) | | | | | $L_2$ (P24) | | | |
|---|---|---|---|---|---|---|---|---|
| Task | Side | Processing Time | Immediate Predecessors | | Task | Side | Processing Time | Immediate Predecessors |
| 1 | E | 6 | - | | 1 | L | 3 | - |
| 2 | E | 5 | - | | 2 | L | 7 | - |
| 3 | L | 2 | 1 | | 3 | R | 7 | - |
| 4 | E | 9 | 1 | | 4 | R | 5 | - |
| 5 | R | 8 | 2 | | 5 | L | 4 | 2 |
| 6 | L | 4 | 3 | | 6 | E | 3 | 2, 3 |
| 7 | E | 7 | 4, 5 | | 7 | R | 4 | 3 |
| 8 | E | 4 | 6, 7 | | 8 | E | 3 | 5 |
| 9 | R | 5 | 7 | | 9 | E | 6 | 6 |
| 10 | R | 4 | 7 | | 10 | E | 4 | 7 |
| 11 | E | 6 | 8 | | 11 | L | 4 | 1 |
| 12 | L | 5 | 9 | | 12 | L | 3 | 8, 9 |
| 13 | E | 6 | 9, 10 | | 13 | E | 3 | 9 |
| 14 | E | 4 | 11 | | 14 | R | 9 | 9, 10 |
| 15 | E | 3 | 11, 12 | | 15 | R | 5 | 4 |
| 16 | E | 4 | 13 | | 16 | L | 9 | 11 |
| - | - | - | - | | 17 | E | 2 | 12 |
| - | - | - | - | | 18 | E | 7 | 13 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| - | - | - | - | | 19 | E | 9 | 13, 14 |
| - | - | - | - | | 20 | R | 9 | 15 |
| - | - | - | - | | 21 | L | 8 | 16, 17 |
| - | - | - | - | | 22 | E | 8 | 18 |
| - | - | - | - | | 23 | R | 9 | 19, 20 |
| - | - | - | - | | 24 | E | 9 | 20 |
| Total Time ($\sum pt_{1i}$): 82 | | | | | Total Time ($\sum pt_{2i}$): 140 | | | |

The lower and upper bounds of the lines are assumed $C_{1LB} = C_{2LB} = 9$ and $C_{1UB} = C_{2UB} = 27$ where $C_{1Inc} = C_{2Inc} = 2$. In real world applications, these bounds could be determined based on demands of the products assembled on the lines however it should be noted that the lower bound cannot be less than the maximum processing time. The reason is that, tasks can be assigned to exactly one workstation and it is not allowed to be split into two or more workstations.

The algorithm is run and the results are recorded for each cycle time combination of the lines. Table 3 reports the obtained results from the first 44 and the last three iterations for the illustration purpose only (where $K$ corresponds to the total number of utilised workstations utilised across the lines; $K = \sum_{h=1}^{H} K_h$). The convergence of the algorithm throughout a hundred iterations is also exhibited in Figure 5.

Table 3. Input data for the numerical example

| # | $C_1$ | $C_2$ | $C$ | $K$ | $SE$ (%) | | # | $C_1$ | $C_2$ | $C$ | $K$ | $SE$ (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9 | 9 | 9 | 30 | **82.2** | | 25 | 13 | 17 | 221 | 18 | 80.7 |
| 2 | 9 | 11 | 99 | 27 | 80.8 | | 26 | 13 | 19 | 247 | 17 | 80.4 |
| 3 | 9 | 13 | 117 | 25 | 79.5 | | 27 | 13 | 21 | 273 | 16 | 81.0 |
| 4 | 9 | 15 | 45 | 24 | 76.8 | | 28 | 13 | 23 | 299 | 16 | 77.4 |
| 5 | 9 | 17 | 153 | 22 | 78.8 | | 29 | 13 | 25 | 325 | 15 | 79.3 |
| 6 | 9 | 19 | 171 | 20 | **82.3** | | 30 | 13 | 27 | 351 | 15 | 76.6 |
| 7 | 9 | 21 | 63 | 20 | 78.8 | | 31 | 15 | 9 | 45 | 25 | 84.0 |
| 8 | 9 | 23 | 207 | 20 | 75.9 | | 32 | 15 | 11 | 165 | 22 | 82.6 |
| 9 | 9 | 25 | 225 | 19 | 77.4 | | 33 | 15 | 13 | 195 | 20 | 81.1 |
| 10 | 9 | 27 | 27 | 17 | **84.0** | | 34 | 15 | 15 | 15 | 19 | 77.8 |
| 11 | 11 | 9 | 99 | 27 | **85.2** | | 35 | 15 | 17 | 255 | 18 | 76.1 |
| 12 | 11 | 11 | 11 | 25 | 80.7 | | 36 | 15 | 19 | 285 | 16 | 80.2 |
| 13 | 11 | 13 | 143 | 22 | 82.8 | | 37 | 15 | 21 | 105 | 16 | 75.8 |
| 14 | 11 | 15 | 165 | 22 | 76.3 | | 38 | 15 | 23 | 345 | 15 | 77.0 |
| 15 | 11 | 17 | 187 | 18 | **87.1** | | 39 | 15 | 25 | 75 | 14 | 79.0 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16 | 11 | 19 | 209 | 18 | 82.3 | | 40 | 15 | 27 | 135 | 14 | 76.0 |
| 17 | 11 | 21 | 231 | 18 | 78.4 | | 41 | 17 | 9 | 153 | 23 | **88.6** |
| 18 | 11 | 23 | 253 | 17 | 79.6 | | 42 | 17 | 11 | 187 | 22 | 79.7 |
| 19 | 11 | 25 | 275 | 17 | 76.7 | | 43 | 17 | 13 | 221 | 21 | 74.2 |
| 20 | 11 | 27 | 297 | 16 | 78.9 | | 44 | 17 | 15 | 255 | 18 | 78.6 |
| 21 | 13 | 9 | 117 | 26 | 84.0 | | … | … | … | … | … | … |
| 22 | 13 | 11 | 143 | 24 | 79.3 | | 98 | 27 | 23 | 621 | 11 | 82.9 |
| 23 | 13 | 13 | 13 | 22 | 77.6 | | 99 | 27 | 25 | 675 | 13 | 66.4 |
| 24 | 13 | 15 | 195 | 20 | 78.2 | | 100 | 27 | 27 | 27 | 10 | 82.2 |

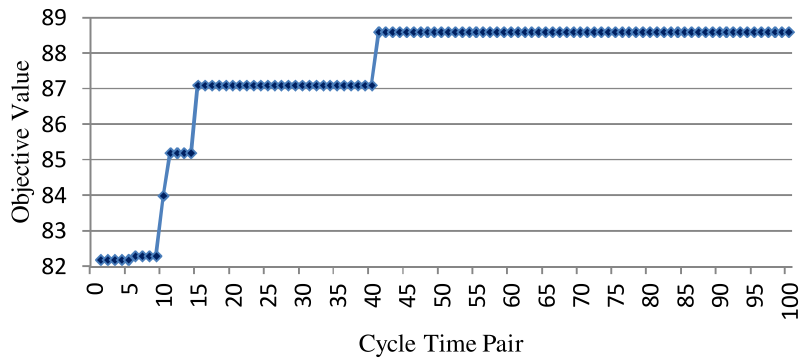


Figure 5. The convergence of the algorithm when multi-line stations are allowed

As could be seen from Table 3 and Figure 5, system efficiency increases gradually in the sixth, tenth, eleventh and fifteenth iterations and reaches its maximum at 88.6% in iteration 41 with 23 workstations, where $C_1 = 17$, $C_2 = 9$ and $C = 153$. The task allocation of the best balancing solution obtained under these circumstances is depicted in Figure 6. As can be seen in the figure, 23 operators are needed to assemble 40 tasks belonging to two different product models. Please note that although tasks with asterisk ($^*$) belong to $L_2$, they are performed from $L_1$ by putting multi-line stations in practice. The efficiency of the configured system could be calculated as $SE = \left[\left((153/17)\cdot 82 + (153/9)\cdot 140\right)/(153\cdot 23)\right]\cdot 100 = 88.6\%$.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $L_1$ | L | 2, 1, 3, 6 | 4, 7 | 8, 11, 14 | 12, 15, 16 | - | - | - |
| | R | 5 | 1*, 11* | 10, 9, 13 | 16* | 24* | 21* | - |
| $L_2$ | L | 2 | 5, 8 | 9, 12 | 13, 10, 17 | 18 | 22 | - |
| | R | 3 | 6, 4 | 15, 7 | 20 | 14 | 19 | 23 |

Figure 6. Task allocation of the obtained best solution when multi-line stations are allowed

If the lines would have been balanced individually (or separately) without the utilisation of multi-line stations, the best solution, which is given in Figure 7, could be obtained in iteration 31 when $C_1 = 15$ and $C_2 = 9$ ($C = 45$). Also, the efficiency of the obtained best solution would be 87.5% (see Figure 8).

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $L_1$ | L | 2, 3, 6 | 7 | 8, 12, 14 | 13, 16, 15 | - | - | - | - | - |
| | R | 1, 4 | 5 | 9, 11, 10 | - | - | - | - | - | - |
| $L_2$ | L | 1, 11 | 16 | 2 | 6, 9 | 24 | 13, 5 | 19 | 8, 12, 17 | 21 |
| | R | 3 | 4, 7 | 15, 10 | 20 | 14 | - | 18 | 23 | 22 |

Figure 7. Task allocation of the obtained best solution when the lines are balanced separately

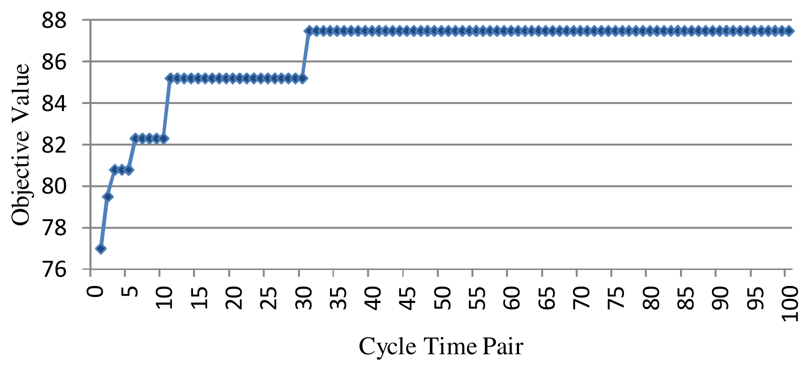


Figure 8. The convergence of the algorithm when the lines are balanced separately

If the solutions obtained *when the lines are balanced together* and *when the lines are balanced separately* are compared, the advantage of utilising multi-line stations could be seen easily. This also proves the practical advantages of parallel two-sided lines over two-sided lines. Multi-line stations help not only increase the system efficiency by minimising the number of workstations, but also minimise the line length as could be seen from the comparison of obtained best solutions. Moreover, shorter line length is the main advantage of parallel two-sided lines over parallel lines as tasks can be performed on both sides of the lines in a parallel two-sided line layout.

## 5. Computational Study

### 5.1. Parameter optimisation

#### 5.1.1. Response surface methodology (RSM)

The parameters of the developed ACO based approach are calibrated using a well-known design of experiment technique, RSM, which has been used extensively in engineering problems. RSM aims at examining and characterising problems where input variables influence some performance aspects of the output (product or process), called *response*. RSM consists of a series of statistical and mathematical techniques used for modelling mathematical relations between the inputs and outputs of a process. It was first proposed by Box and Wilson (Box and Wilson, 1951) for the aim of determining the optimum combination of factors, which minimises the output of a real non-simulated system (Dhupal *et al.*, 2007; Hossein Safizadeh and Thornton, 1984).

RSM consumes less time and effort in comparison with trying all combinations of the parameters one-by-one, which needs much time and costs more. In RSM, numerous factors are tested simultaneously in a limited number of experiments, for product or process optimisation. Also, it is possible to quantitatively measure possible interactions between factors, important information which is hardly possible to obtain using other optimisation techniques (Bayhan and Onel, 2010; Kucukkoc *et al.*, 2013). The form of the relationship between independent variables and the response is unknown and approximated in most cases. The general second-order polynomial response surface mathematical model (full quadratic model) for the experimental design is given in Equation (10) (Dhupal *et al.*, 2007; Yalcinkaya and Bayhan, 2009).

$$Y_u = \beta_0 + \sum_{i=1}^{n} \beta_i X_{iu} + \sum_{i=1}^{n} \beta_{ii} X_{iu}^2 + \sum_{i<j}^{n} \beta_{ij} X_{iu} X_{ju} + e_u \tag{10}$$

where $Y_u$ is the corresponding response; $\beta_0$, $\beta_i$, $\beta_{ii}$ and $\beta_{ij}$ represent the regression coefficients; $X_{iu}$ and $X_{ju}$ are coded values of the $i^{th}$ and $j^{th}$ input parameters $(i < j)$ respectively, and $e_u$ is the residual experimental error of the $u^{th}$ observation.

The model in terms of the observations may be written in matrix notation as $Y = \beta X + \varepsilon$, where $Y$ and $X$ represent output and input matrices, respectively; and $\varepsilon$ is the matrix of residuals (error term) (Montgomery, 2001). The least square estimator of $\beta$ matrix that is composed of coefficients of the regression equation is calculated as $\beta = (X'X)^{-1}X'Y$ (Kucukkoc *et al.*, 2013). The fitted regression models with the coefficients for fitness value are formulated in the next section.

### 5.1.2. Parameter optimisation of ACO algorithm

The ACO parameters - namely $\alpha$, $\beta$, $\rho$ and colony size $(\sigma)$ - are optimised through

Minitab™17 statistical software, which uses the mathematical structure given in Section 5.1. The considered factor levels of the considered parameters for the experiments are given in Table 4. In determining these factors and their levels, similar studies, which proposed ACO algorithm to tackle line balancing problems, have been referenced. As the efficiency of the obtained solution is maximised, the *Average System Efficiency (ASE)* and the *Best System Efficiency (BSE)* values will be considered as response. *ASE* is obtained through dividing sum of all obtained *SE* values by total number of solutions (equivalent to number of ants), while *BSE* is the global best *SE* value.

Table 4. Levels and values of the ACO parameters

| Parameter | Symbol | Level | | | | |
|---|---|---|---|---|---|---|
| | | -2 | -1 | 0 | 1 | 2 |
| Pheromone Constant | $\alpha$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
| Heuristic Constant | $\beta$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
| Evaporation Rate | $\rho$ | 0.05 | 0.2 | 0.35 | 0.5 | 0.65 |
| Colony Size | $\sigma$ | 10 | 15 | 20 | 25 | 30 |

Experiments are accomplished on a randomly selected test case (test case #11) given in Section 5.2, according to the experimental design given in Table 5 with uncoded values of factors and run orders. The ACO algorithm is run with the designated factor levels for each experiment considered and the responses are reported in Table 5.

Table 5. Design of experiments matrix showing uncoded values of factors and observed responses

| Experiment No. | Run Order | | Uncoded Value | | | | | Responses | |
|---|---|---|---|---|---|---|---|---|---|
| | | | $\alpha$ | $\beta$ | $\rho$ | $\sigma$ | | *ASE* | *BSE* |
| 1 | 1 | | 0.3 | 0.3 | 0.20 | 15 | | 77.2 | 94.2 |
| 2 | 2 | | 0.7 | 0.3 | 0.20 | 15 | | 77.5 | 91.6 |
| 3 | 3 | | 0.3 | 0.7 | 0.20 | 15 | | 77.3 | 91.6 |
| 4 | 4 | | 0.7 | 0.7 | 0.20 | 15 | | 77.4 | 91.5 |
| 5 | 5 | | 0.3 | 0.3 | 0.50 | 15 | | 77.3 | 91.6 |
| 6 | 6 | | 0.7 | 0.3 | 0.50 | 15 | | 77.3 | 89.6 |
| 7 | 7 | | 0.3 | 0.7 | 0.50 | 15 | | 77.5 | 92.5 |
| 8 | 8 | | 0.7 | 0.7 | 0.50 | 15 | | 77.3 | 90.1 |
| 9 | 9 | | 0.3 | 0.3 | 0.20 | 25 | | 77.4 | 91.6 |
| 10 | 10 | | 0.7 | 0.3 | 0.20 | 25 | | 77.3 | 91.6 |
| 11 | 11 | | 0.3 | 0.7 | 0.20 | 25 | | 77.3 | 91.5 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 12 | | 0.7 | 0.7 | 0.20 | 25 | | 77.4 | 94.2 |
| 13 | 13 | | 0.3 | 0.3 | 0.50 | 25 | | 77.4 | 91.7 |
| 14 | 14 | | 0.7 | 0.3 | 0.50 | 25 | | 77.4 | 92.1 |
| 15 | 15 | | 0.3 | 0.7 | 0.50 | 25 | | 77.4 | 91.6 |
| 16 | 16 | | 0.7 | 0.7 | 0.50 | 25 | | 77.5 | 91.6 |
| 17 | 17 | | 0.1 | 0.5 | 0.35 | 20 | | 77.3 | 91.5 |
| 18 | 18 | | 0.9 | 0.5 | 0.35 | 20 | | 77.3 | 90.1 |
| 19 | 19 | | 0.5 | 0.1 | 0.35 | 20 | | 77.3 | 90.1 |
| 20 | 20 | | 0.5 | 0.9 | 0.35 | 20 | | 77.5 | 92.5 |
| 21 | 21 | | 0.5 | 0.5 | 0.05 | 20 | | 77.2 | 91.5 |
| 22 | 22 | | 0.5 | 0.5 | 0.65 | 20 | | 77.4 | 91.6 |
| 23 | 23 | | 0.5 | 0.5 | 0.35 | 10 | | 77.4 | 91.6 |
| 24 | 24 | | 0.5 | 0.5 | 0.35 | 30 | | 77.4 | 91.9 |
| 25 | 25 | | 0.5 | 0.5 | 0.35 | 20 | | 77.3 | 94.2 |
| 26 | 26 | | 0.5 | 0.5 | 0.35 | 20 | | 77.4 | 91.5 |
| 27 | 27 | | 0.5 | 0.5 | 0.35 | 20 | | 77.5 | 92.1 |
| 28 | 28 | | 0.5 | 0.5 | 0.35 | 20 | | 77.4 | 91.6 |
| 29 | 29 | | 0.5 | 0.5 | 0.35 | 20 | | 77.2 | 92.6 |
| 30 | 30 | | 0.5 | 0.5 | 0.35 | 20 | | 77.3 | 91.7 |
| 31 | 31 | | 0.5 | 0.5 | 0.35 | 20 | | 77.3 | 92.1 |

Regression equations, which depict the RSM based mathematical models that represent the relations between the responses ($ASE$ and $BSE$) and the factors ($\alpha, \beta, \rho$ and $\sigma$) based on the observed results, are given in Equations (11) and (12) in uncoded units.

$$\begin{aligned} ASE = 77.357 &+ 0.820\,\alpha - 0.305\,\beta + 0.139\,\rho - 0.0279\,\sigma - 0.190\,\alpha^2 + 0.435\,\beta^2 - 0.337\,\rho^2 \\ &+ 0.000696\,\sigma^2 - 0.156\,\alpha * \beta - 1.042\,\alpha * \rho - 0.0063\,\alpha * \sigma + 0.625\,\beta * \rho \\ &- 0.0063\,\beta * \sigma + 0.0250\,\rho * \sigma \end{aligned} \tag{11}$$

$$\begin{aligned} BSE = 99.01 &- 7.40\,\alpha - 2.88\,\beta - 0.9\,\rho - 0.394\,\sigma - 7.28\,\alpha^2 - 4.16\,\beta^2 - 4.62\,\rho^2 - 0.00215\,\sigma^2 \\ &+ 6.88\,\alpha * \beta - 8.33\,\alpha * \rho + 0.638\,\alpha * \sigma + 2.08\,\beta * \rho + 0.200\,\beta * \sigma + 0.267\,\rho * \sigma \end{aligned} \tag{12}$$

When the parameter optimisation is performed with the aim of maximising $ASE$ and $BSE$ values, optimal uncoded process parameter settings for the ACO algorithm are achieved as $\alpha = 0.7222$, $\beta = 0.90$, $\rho = 0.4621$ and $\sigma = 30$ with composite desirability of $d = 1$ (see Figure 9).

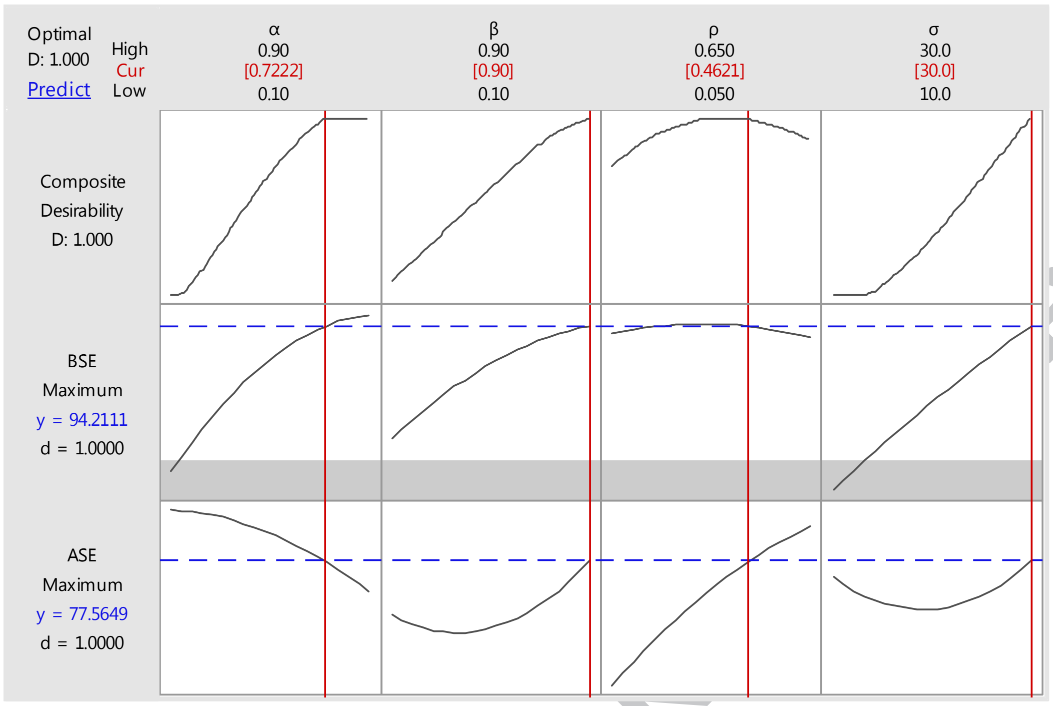


Figure 9. Optimisation results for ACO parameters

Next section presents the experimental tests, which are conducted using the optimised values of the above parameters for each test case.

## 5.2. Experimental Tests

### 5.2.1. Input data

The proposed algorithm was coded in Java SE 7u4 environment and run on a 3.1 GHz Intel Core i5-2400 CPU 4GB RAM computer using the calibrated parameters to test its performance. The same test problems with Ozcan *et al.* (2010b), which were derived from the literature for the type-1 parallel two-sided assembly line balancing problem, are solved in different combinations (where each combination is called *test case*) using the developed ACO based algorithm in this research as well. Therefore, original test problems P9, P12 and P24 are taken from Kim *et al.* (2000); P16, A65 and A205 are taken from Lee *et al.* (2001); and

B148 is taken from Bartholdi (1993) (B148 was then modified by Lee *et al*. (2001)[*]) to test the performance of the tabu search algorithm in solving two-sided assembly line balancing problems.

Table 6 presents input data for the computational study carried out and shows which problem is considered on which line for each of the test cases. As mentioned, a test case corresponds to a pair of two test problems, one on each of the parallel two-sided lines. For example, in test case #9, P16 is accommodated on $L_1$ while P24 is performed on $L_2$. As could be seen in the table, new test cases are also added to the experimental dataset, *i.e.* test cases #3, #4, #7, #10, #13, #16 and #19, in addition to the test cases solved by Ozcan *et al*. (2010b). In the table, $\sum T_{hi}$ and $\sum pt_{hi}$ columns give *total number of tasks* and *sum of task processing times*, respectively, for each problem considered on the particular line. The maximum task processing times, represented with $max\ (pt_{hi})$; lower bounds ($C_{hLB}$), upper bounds ($C_{hUB}$) and increments ($C_{hInc}$) of cycle times are also reported for each problem. It should be noted here that no constant cycle times are given in the table as this study also endeavours to minimise the cycle time as well as the total number of workstations.

[*] As the original values are much larger than the others, Lee *et al*. (2001) changed the processing times of tasks 79 and 108 from 281 to 111 and from 383 to 43, respectively.

Table 6. Input data of the test cases designed for the computational tests

| Test Case | | Line I ($L_1$) | | | | | | | | | Line II ($L_2$) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Problem | $\sum T_{1i}$ | $\sum pt_{1i}$ | $max\ (pt_{1i})$ | $C_{1LB}$ | $C_{1UB}$ | $C_{1Inc}$ | | | Problem | $\sum T_{2i}$ | $\sum pt_{2i}$ | $max\ (pt_{2i})$ | $C_{2LB}$ | $C_{2UB}$ | $C_{2Inc}$ |
| 1 | | P9 | 9 | 17 | 3 | 3 | 9 | 1 | | | P9 | 9 | 17 | 3 | 3 | 9 | 1 |
| 2 | | P9 | 9 | 17 | 3 | 3 | 9 | 1 | | | P12 | 12 | 25 | 3 | 3 | 9 | 1 |
| 3 | | P9 | 9 | 17 | 3 | 3 | 9 | 1 | | | P16 | 16 | 82 | 9 | 9 | 27 | 2 |
| 4 | | P12 | 12 | 25 | 3 | 3 | 9 | 1 | | | P9 | 9 | 17 | 3 | 3 | 9 | 1 |
| 5 | | P12 | 12 | 25 | 3 | 3 | 9 | 1 | | | P12 | 12 | 25 | 3 | 3 | 9 | 1 |
| 6 | | P12 | 12 | 25 | 3 | 3 | 9 | 1 | | | P16 | 16 | 82 | 9 | 9 | 27 | 2 |
| 7 | | P16 | 16 | 82 | 9 | 9 | 27 | 2 | | | P12 | 12 | 25 | 3 | 3 | 9 | 1 |
| 8 | | P16 | 16 | 82 | 9 | 9 | 27 | 2 | | | P16 | 16 | 82 | 9 | 9 | 27 | 2 |
| 9 | | P16 | 16 | 82 | 9 | 9 | 27 | 2 | | | P24 | 24 | 140 | 9 | 9 | 27 | 2 |
| 10 | | P24 | 24 | 140 | 9 | 9 | 27 | 2 | | | P16 | 16 | 82 | 9 | 9 | 27 | 2 |
| 11 | | P24 | 24 | 140 | 9 | 9 | 27 | 2 | | | P24 | 24 | 140 | 9 | 9 | 27 | 2 |
| 12 | | P24 | 24 | 140 | 9 | 9 | 27 | 2 | | | A65 | 65 | 5099 | 272 | 326 | 816 | 32 |
| 13 | | A65 | 65 | 5099 | 272 | 326 | 816 | 32 | | | P24 | 24 | 140 | 9 | 9 | 27 | 2 |
| 14 | | A65 | 65 | 5099 | 272 | 326 | 816 | 32 | | | A65 | 65 | 5099 | 272 | 326 | 816 | 32 |
| 15 | | A65 | 65 | 5099 | 272 | 326 | 816 | 32 | | | B148 | 148 | 5024 | 170 | 204 | 680 | 32 |
| 16 | | B148 | 148 | 5024 | 170 | 204 | 680 | 32 | | | A65 | 65 | 5099 | 272 | 326 | 816 | 32 |
| 17 | | B148 | 148 | 5024 | 170 | 204 | 680 | 32 | | | B148 | 148 | 5024 | 170 | 204 | 680 | 32 |
| 18 | | B148 | 148 | 5024 | 170 | 204 | 680 | 32 | | | A205 | 205 | 23345 | 944 | 1510 | 3776 | 128 |
| 19 | | A205 | 205 | 23345 | 944 | 1510 | 3776 | 128 | | | B148 | 148 | 5024 | 170 | 204 | 680 | 32 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | | A205 | 205 | 23345 | 944 | 1510 | 3776 | 128 | | | A205 | 205 | 23345 | 944 | 1510 | 3776 | 128 |

### 5.2.2. Test results

For each problem, cycle times calculated by the algorithm between the designated lower and upper bounds ($C_{hLB}$ and $C_{hUB}$) in accordance with the increments ($C_{hInc}$) given in Table 6 are presented in Table 7. To explain, if we consider the test case #15 in Table 6, where problems A65 and B148 are utilised on $L_1$ and $L_2$, respectively, ACO algorithm will try to build a balancing solution for $C_1 = 326$ and $C_2 = 204$ first (where $C = LCM(326, 204) = 33252$). After that, $C_2$ will be increased by 32 units (will reach to 236) and a new balancing solution will be sought when $C_1 = 326$, $C_2 = 236$ and $C = 38468$; and so on. After all possible values of $C_2$ are tried one-by-one while $C_1$ remains the same ($C_1 = 326$), $C_1$ is increased by 32 units ($C_1 = 358$) and all possible combinations of $C_2$ are tried again one-by-one. This cycle continues until all possible combinations of $C_1$ and $C_2$ are tried. Total number of all possible combinations, represented by $TPC$, could be calculated by multiplying *total number of possible $C_1$ values* by *total number of possible $C_2$ values: $card(C_1) \times card(C_2)$,* where $card(C_h)$ represents total number of possible cycle time values for line $L_h$.

Table 7. Calculated cycle time values for the considered problems

| Problem | Cycle Times |
|---|---|
| P9 | 3, 4, 5, 6, 7, 8, 9 |
| P12 | 3, 4, 5, 6, 7, 8, 9 |
| P16 | 9, 11, 13, 15, 17, 19, 21, 23, 25, 27 |
| P24 | 9, 11, 13, 15, 17, 19, 21, 23, 25, 27 |
| A65 | **326**, 358, 390, 422, 454, 486, 518, 550, 582, 614, 646, 678, 710, 742, 774, 806 |
| B148 | **204**, **236**, 268, 300, 332, 364, 396, 428, 460, 492, 524, 556, 588, 620, 652 |
| A205 | 1510, 1638, 1766, 1894, 2022, 2150, 2278, 2406, 2534, 2662, 2790, 2918, 3046, 3174, 3302, 3430, 3558, 3686 |

The ACO algorithm is run using the parameters obtained through the RSM and the best solution is taken after one run for each test problem. As there is no comparable result in the literature (due to the fact that the PTALBP-E has never been addressed by any researcher in the literature so far), same test cases are also solved using three other well-known heuristics for the comparison purpose: *(i) Longest Processing Time – LPT (Talbot and Patterson, 1984), (ii) COMSOAL (Arcus, 1966),* and *(iii) Maximum Number of Immediate Successors – MNIS (Tonge, 1960)*. Each of the three heuristics is run for five times for each test case. Table 8 reports the computational results of 20 test cases obtained through LPT, COMSOAL, MNIS and ACO algorithms. In the table, $TPC$ corresponds to the total number of all possible combinations of $C_1$ and $C_2$ values (to remind a new cycle time pair is tried in each iteration). In the columns $C_1$, $C_2$, $C$ and $K$; the cycle time found for $L_1$, the cycle time found for $L_2$, the common cycle time and the total number of workstations belonging to the best solution are given. For each test case, the average system efficiency $(ASE)$ and the best system efficiency $(BSE)$ values of the obtained solutions are also reported in the table.

As could be seen from the results table, quite high $BSE$ values are obtained by ACO for the majority of the test cases solved. The maximum $BSE$ value, which is 97.6, is reported for the test case #4; while the minimum $BSE$ value, which is 81.3, is observed for the test case #20 where a total of 410 tasks are balanced. As the number of tasks increases, the problem size grows exponentially; and the larger the problem size, the more effort needed for the solution approach. This is why $ASE$ and $BSE$ values are slightly reduced when the problem size increases.

It can be seen that the BSE values found by ACO are equal to the BSE values obtained by other test heuristics (LPT, COMSOAL and MNIS) for only two test cases, *i.e.* #1 and #5. LPT and MNIS find the same BSE value (*i.e.* 94.4) with ACO for test case #1 while only MNIS finds the same BSE value (*i.e.* 92.5) with ACO for test case #5. For all of the remaining 18 test cases, ACO finds better solutions than any of the other heuristics. That means none of the test heuristics can find solutions equal to or better than ACO when the problem size increases.

Table 8. Representation of the obtained solutions for the solved test cases

| # | TP | LPT | | | | | | | COMSOAL | | | | | | | MNIS | | | | | | | ACO | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | $C_1$ | $C_2$ | $C$ | $K$ | $AS$ | $BS$ | | $C_1$ | $C_2$ | $C$ | $K$ | $AS$ | $BS$ | | $C_1$ | $C_2$ | $C$ | $K$ | $AS$ | $BS$ | | $C_1$ | $C_2$ | $C$ | $K$ | $AS$ | $BS$ | |
| *1* | 49 | 6 | 6 | 6 | 6 | 72. | **94.** | | 8 | 5 | 40 | 6 | 71. | 92. | | 6 | 9 | 18 | 5 | 72. | **94.** | | 6 | 6 | 6 | 6 | 71. | **94.** | |
| *2* | 49 | 9 | 3 | 9 | 1 | 73. | 92. | | 3 | 3 | 3 | 1 | 70. | 93. | | 3 | 3 | 3 | 1 | 72. | 93. | | 4 | 3 | 12 | 1 | 71. | **96.** | |
| *3* | 70 | 3 | 23 | 69 | 1 | 67. | 83. | | 5 | 11 | 55 | 1 | 66. | 83. | | 3 | 17 | 51 | 1 | 66. | 87. | | 6 | 23 | 138 | 7 | 65. | **91.** | |
| *4* | 49 | 3 | 5 | 15 | 1 | 73. | 90. | | 3 | 9 | 9 | 1 | 71. | 92. | | 5 | 3 | 15 | 1 | 72. | 88. | | 8 | 3 | 24 | 9 | 72. | **97.** | |
| *5* | 49 | 4 | 3 | 12 | 1 | 74. | 85. | | 3 | 8 | 24 | 1 | 73. | 88. | | 3 | 3 | 3 | 1 | 73. | **92.** | | 3 | 3 | 3 | 1 | 73. | **92.** | |
| *6* | 70 | 3 | 9 | 9 | 2 | 68. | 87. | | 3 | 15 | 15 | 1 | 67. | 86. | | 3 | 15 | 15 | 1 | 68. | 86. | | 3 | 23 | 69 | 1 | 68. | **91.** | |
| *7* | 70 | 17 | 3 | 51 | 1 | 69. | 87. | | 21 | 3 | 21 | 1 | 67. | 87. | | 15 | 3 | 15 | 1 | 68. | 86. | | 25 | 3 | 75 | 1 | 68. | **89.** | |
| *8* | 10 | 9 | 23 | 207 | 1 | 65. | 84. | | 11 | 11 | 11 | 1 | 63. | 82. | | 11 | 11 | 11 | 1 | 64. | 78. | | 15 | 17 | 255 | 1 | 64. | **85.** | |
| *9* | 10 | 19 | 9 | 171 | 2 | 73. | 86. | | 25 | 21 | 525 | 1 | 72. | 82. | | 21 | 9 | 63 | 2 | 73. | 88. | | 23 | 25 | 575 | 1 | 72. | **91.** | |
| *1* | 10 | 9 | 27 | 27 | 2 | 73. | 84. | | 19 | 17 | 323 | 1 | 71. | 87. | | 17 | 15 | 255 | 1 | 73. | 85. | | 9 | 15 | 45 | 2 | 72. | **91.** | |
| *1* | 10 | 9 | 21 | 63 | 2 | 79. | 88. | | 27 | 19 | 513 | 1 | 77. | 89. | | 9 | 9 | 9 | 3 | 77. | 88. | | 9 | 25 | 225 | 2 | 77. | **91.** | |
| *1* | 16 | 11 | 486 | 5346 | 2 | 76. | 89. | | 11 | 486 | 5346 | 2 | 75. | 85. | | 9 | 678 | 2034 | 2 | 76. | 88. | | 17 | 454 | 7718 | 2 | 75. | **92.** | |
| *1* | 16 | 454 | 9 | 4086 | 3 | 73. | 89. | | 390 | 9 | 1170 | 3 | 71. | 84. | | 358 | 9 | 3222 | 3 | 72. | 87. | | 710 | 9 | 6390 | 2 | 75. | **90.** | |
| *1* | 25 | 518 | 742 | 2745 | 1 | 74. | 87. | | 326 | 326 | 326 | 3 | 73. | 84. | | 582 | 422 | 12280 | 2 | 75. | 86. | | 358 | 486 | 8699 | 2 | 74. | **88.** | |
| *1* | 24 | 486 | 492 | 3985 | 2 | 73. | 86. | | 390 | 204 | 13260 | 4 | 73. | 85. | | 326 | 268 | 43684 | 3 | 75. | 88. | | 358 | 300 | 5370 | 3 | 73. | **88.** | |
| *1* | 24 | 428 | 326 | 6976 | 3 | 73. | 85. | | 332 | 390 | 64740 | 3 | 72. | 85. | | 268 | 358 | 47972 | 3 | 75. | 86. | | 300 | 358 | 5370 | 3 | 73. | **88.** | |
| *1* | 22 | 204 | 236 | 1203 | 5 | 73. | 85. | | 204 | 236 | 12036 | 5 | 73. | 86. | | 236 | 204 | 12036 | 5 | 76. | 88. | | 236 | 300 | 1770 | 4 | 73. | **90.** | |
| *1* | 27 | 300 | 215 | 1290 | 3 | 65. | 78. | | 300 | 163 | 81900 | 3 | 66. | 81. | | 268 | 151 | 20234 | 4 | 70. | 83. | | 268 | 163 | 2194 | 3 | 67. | **84.** | |
| *1* | 27 | 266 | 204 | 2715 | 4 | 66. | 81. | | 151 | 204 | 15402 | 4 | 67. | 81. | | 151 | 268 | 20234 | 4 | 66. | 81. | | 151 | 236 | 1781 | 4 | 68. | **83.** | |
| *2* | 32 | 163 | 163 | 1638 | 3 | 58. | 75. | | 151 | 176 | 13333 | 3 | 60. | 75. | | 151 | 163 | 12366 | 3 | 63. | 78. | | 151 | 151 | 1510 | 3 | 60. | **81.** | |

Moreover, the maximum BSE value found by any of the test heuristics for any of the solved test cases is 94.4 while the minimum is 75.0, for test cases #1 and #20, respectively. It is clear that the maximum and the minimum BSE values obtained by ACO (*i.e.* 97.6 and 81.3, respectively) are far beyond the values obtained by the test heuristics. Therefore, it could be said that ACO outperforms LPT, COMSOAL and MNIS in terms of solved test cases within the scope of this research.

## 6. Conclusions

Minimisation of cycle time and minimisation of number of workstations are two major objectives considered separately in most of the line balancing problems. Although these two objectives conflict with each other, minimisation of both performance measures at the same time has a significant effect on the efficiency of the entire production system from a managerial point of view.

The main contribution of this paper is to aim at minimising these two conflicting objectives for a recently introduced line configuration, which is parallel two-sided assembly line system, for the first time in the literature. As each of the parallel lines may have a different cycle time, every possible combination of the cycle times (called cycle time pair) is investigated within the determined lower and upper bounds and the lines are balanced for every new situation. The solution which gives the best system efficiency value is designated as the global best solution of the problem.

Another significant contribution of the paper is that a new ACO based approach, where ACO parameters are optimised via RSM, is proposed as a possible solution approach for balancing parallel two-sided assembly line systems for the first time. Moreover, to the best of the authors' knowledge, this is the unique study that applies RSM to optimise ACO parameters in the overall assembly line balancing domain.

A numerical example is provided; *(i)* to present the running principle of the proposed approach, *(ii)* to show the benefits of constructing multi-line stations, and *(iii)* to clarify the advantages of the handled line configuration over parallel or two-sided line systems. A set of test problems, which were originally derived for type-1 PTALBP, are solved using the proposed approach and three other well-known heuristics with the aim of minimisation both cycle time and total number of workstations on parallel two-sided assembly lines. The experimental results indicate that the proposed ACO algorithm outperforms other three well known heuristics (namely LPT, COMSOAL and MNIS) used for comparison purposes. Although the complexity of the problem is higher than other configurations of assembly lines (*i.e.* one-sided straight assembly lines, two-sided assembly lines, *etc.*), well balanced solutions are observed and reported in the research to establish a base point for future researches and to provide test results for comparison purposes.

Developing mathematical formulation of the problem and new solution methods (exact, heuristic and meta-heuristic approaches) could be considered in future studies. In addition, some other techniques could also be hybridised with the ACO based approach proposed in this research.

## References

Akpinar, S., Bayhan, G.M., Baykasoglu, A., 2013. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Applied Soft Computing, 13(1), 574-589.
Arcus, A., 1966. COMSOAL, A Computer Method of Sequencing Operations for Assembly Lines. The International Journal of Production Research, 4(4), 259-277.
Avikal, S., Jain, R., Mishra, P.K., Yadav, H.C., 2013. A heuristic approach for U-shaped assembly line balancing to improve labor productivity. Computers & Industrial Engineering, 64(4), 895-901.
Balseiro, S.R., Loiseau, I., Ramonet, J., 2011. An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows. Computers & Operations Research, 38(6), 954-966.

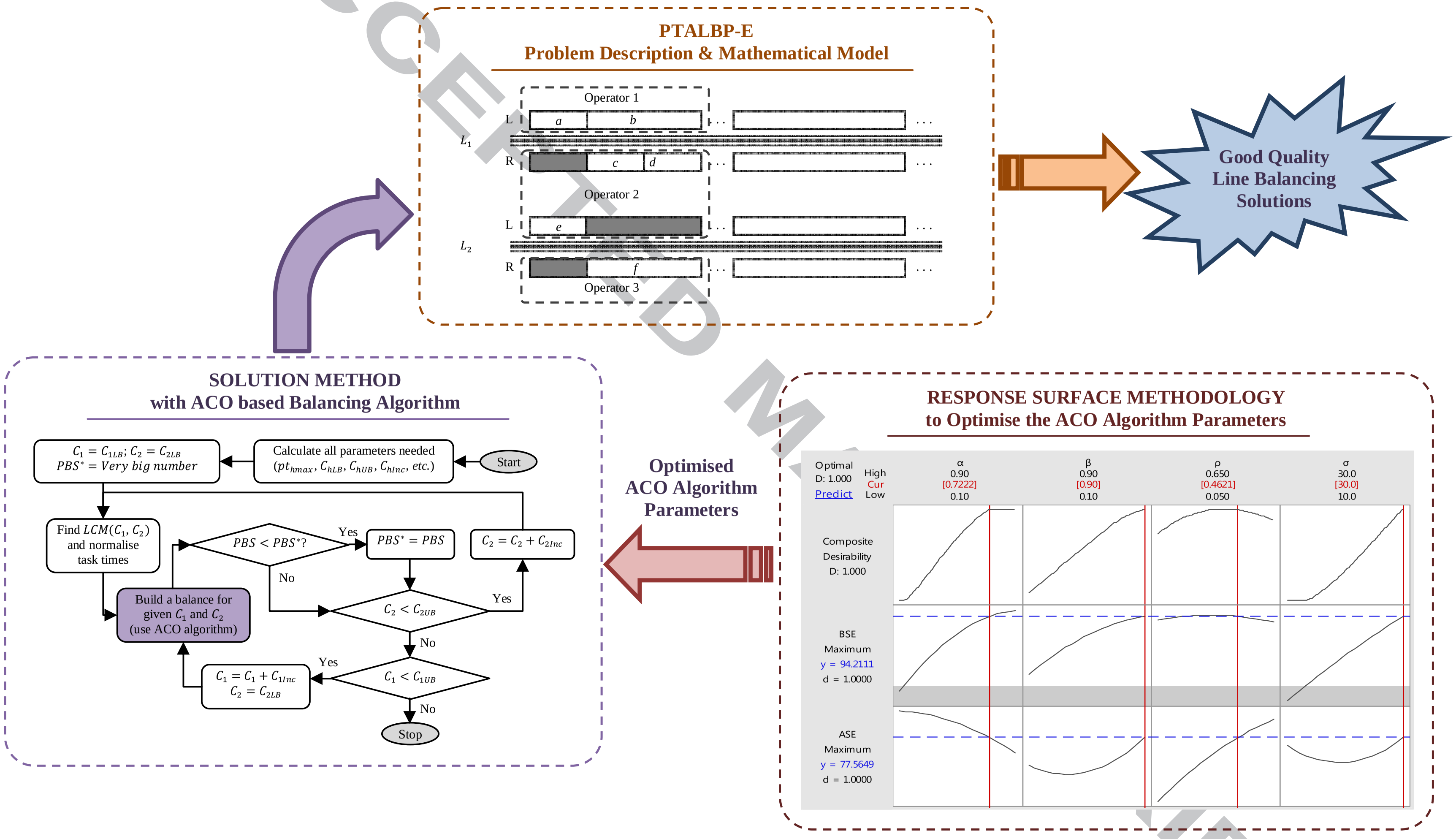
Bartholdi, J.J., 1993. Balancing 2-Sided Assembly Lines - a Case-Study. International Journal of Production Research, 31(10), 2447-2461.
Bayhan, M., Onel, K., 2010. Optimization of reinforcement content and sliding distance for AlSi7Mg/SiCp composites using response surface methodology. Materials & Design, 31(6), 3015-3022.
Baykasoglu, A., 2006. Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. Journal of Intelligent Manufacturing, 17(2), 217-232.
Baykasoglu, A., Dereli, T., 2008. Two-sided assembly line balancing using an ant-colony-based heuristic. International Journal of Advanced Manufacturing Technology, 36(5-6), 582-588.
Baykasoglu, A., Ozbakir, L., Gorkemli, L., Gorkemli, B., 2009. Balancing Parallel Assembly Lines via Ant Colony Optimization. Cie: 2009 International Conference on Computers and Industrial Engineering, Vols 1-3506-511.
Benzer, R., Gokcen, H., Cetinyokus, T., Cercioglu, H., 2007. A network model for parallel line balancing Problem. Mathematical Problems in Engineering, doi:10.1155/2007/10106.
Blum, C., 2008. Beam-ACO for Simple Assembly Line Balancing. Informs Journal on Computing, 20(4), 618-627.
Box, G.E.P., Wilson, K.B., 1951. On the Experimental Attainment of Optimum Conditions. Journal of the Royal Statistical Society Series B-Statistical Methodology, 13(1), 1-45.
Cercioglu, H., Ozcan, U., Gokcen, H., Toklu, B., 2009. A Simulated Annealing Approach for Parallel Assembly Line Balancing Problem. Journal of the Faculty of Engineering and Architecture of Gazi University, 24(2), 331-341.
Chen, F., Wang, H., Qi, C., Xie, Y., 2013. An ant colony optimization routing algorithm for two order pickers with congestion consideration. Computers & Industrial Engineering, 66(1), 77-85.
Chen, S.-M., Chien, C.-Y., 2011. Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Systems with Applications, 38(4), 3873-3883.
Cheng, C.-B., Mao, C.-P., 2007. A modified ant colony system for solving the travelling salesman problem with time windows. Mathematical and Computer Modelling, 46(9-10), 1225-1235.
Chutima, P., Chimklai, P., 2012. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. Computers & Industrial Engineering, 62(1), 39-55.
Deng, G.-F., Lin, W.-T., 2011. Ant colony optimization-based algorithm for airline crew scheduling problem. Expert Systems with Applications, 38(5), 5787-5793.
Dhupal, D., Doloi, B., Bhattacharyya, B., 2007. Optimization of process parameters of Nd : YAG laser microgrooving of Al2TiO5 ceramic material by response surface methodology and artificial neural network algorithm. Proceedings of the Institution of Mechanical Engineers Part B-Journal of Engineering Manufacture, 221(8), 1341-1351.
Dorigo, M., Di Caro, G., Gambardella, L.M., 1999. Ant Algorithms for Discrete Optimization. Artificial Life, 5137–172.
Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. Biosystems, 43(2), 73-81.
Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: Optimization by a colony of cooperating agents. Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics, 26(1), 29-41.
Escario, J.B., Jimenez, J.F., Giron-Sierra, J.M., 2015. Ant Colony Extended: Experiments on the Travelling Salesman Problem. Expert Systems with Applications, 42(1), 390-410.
Fattahi, P., Roshani, A., Roshani, A., 2011. A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. International Journal of Advanced Manufacturing Technology, 53(1-4), 363-378.
Gökçen, H., Agpak, K., Benzer, R., 2006. Balancing of parallel assembly lines. International Journal of Production Economics, 103(2), 600-609.
Helgeson, W.B., Birnie, D.P., 1961. Assembly Line Balancing Using the Ranked Positional Weight Technique. The Journal of Industrial Engineering, 12(6), 394-398.

Hossein Safizadeh, M., Thornton, B.M., 1984. Optimization in simulation experiments using response surface methodology. Computers & Industrial Engineering, 8(1), 11-27.
Hu, X.F., Wu, E.F., Bao, J.S., Jin, Y., 2010. A branch-and-bound algorithm to minimize the line length of a two-sided assembly line. European Journal of Operational Research, 206(3), 703-707.
Hu, X.F., Wu, E.F., Jin, Y., 2008. A station-oriented enumerative algorithm for two-sided assembly line balancing. European Journal of Operational Research, 186(1), 435-440.
Kara, Y., Gokcen, H., Atasagun, Y., 2010. Balancing parallel assembly lines with precise and fuzzy goals. International Journal of Production Research, 48(6), 1685-1703.
Kara, Y., Ozguven, C., Yalcin, 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.
Khorasanian, D., Hejazi, S.R., Moslehi, G., 2013. Two-sided assembly line balancing considering the relationships between tasks. Computers & Industrial Engineering, 66(4), 1096-1105.
Kim, Y.K., Kim, Y.H., Kim, Y.J., 2000. Two-sided assembly line balancing: a genetic algorithm approach. Production Planning & Control, 11(1), 44-53.
Kim, Y.K., Song, W.S., Kim, J.H., 2009. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers & Operations Research, 36(3), 853-865.
Kucukkoc, I., Karaoglan, A.D., Yaman, R., 2013. Using response surface design to determine the optimal parameters of genetic algorithm and a case study. International Journal of Production Research, 51(17), 5039-5054.
Kucukkoc, I., Zhang, D.Z., 2014a. An Agent Based Ant Colony Optimisation Approach for Mixed-Model Parallel Two-Sided Assembly Line Balancing Problem, in: Grubbström, R.W., Hinterhuber, H.H. (Eds.), Pre-Prints of the Eighteenth International Working Seminar on Production Economics. Congress Innsbruck, Innsbruck, Austria, pp. 313-328.
Kucukkoc, I., Zhang, D.Z., 2014b. Mathematical Model and Agent Based Solution Approach for the Simultaneous Balancing and Sequencing of Mixed-Model Parallel Two-Sided Assembly Lines. International Journal of Production Economics, 158, 314-333.
Kucukkoc, I., Zhang, D.Z., 2014c. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Research, 52(12), 3665-3687.
Lee, T.O., Kim, Y., Kim, Y.K., 2001. Two-sided assembly line balancing to maximize work relatedness and slackness. Computers & Industrial Engineering, 40(3), 273-292.
Leung, C.W., Wong, T.N., Mak, K.L., Fung, R.Y.K., 2010. Integrated process planning and scheduling by an agent-based ant colony optimization. Computers & Industrial Engineering, 59(1), 166-180.
Lusa, A., 2008. A survey of the literature on the multiple or parallel assembly line balancing problem. European Journal of Industrial Engineering, 2(1), 50-72.
Mavrovouniotis, M., Yang, S., 2013. Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. Applied Soft Computing, 13(10), 4023-4037.
McMullen, P.R., Tarasewich, P., 2003. Using ant techniques to solve the assembly line balancing problem. Iie Transactions, 35(7), 605-617.
McMullen, P.R., Tarasewich, P., 2006. Multi-objective assembly line balancing via a modified ant colony optimization technique. International Journal of Production Research, 44(1), 27-42.
Montgomery, D.C., 2001. Design and Analysis of Experiments, 5th ed. . John Wiley, New York.
Ozbakir, L., Baykasoglu, A., Gorkemli, B., Gorkemli, L., 2011. Multiple-colony ant algorithm for parallel assembly line balancing problem. Applied Soft Computing, 11(3), 3186-3198.
Ozbakir, L., Tapkan, P., 2010. Balancing fuzzy multi-objective two-sided assembly lines via Bees Algorithm. Journal of Intelligent & Fuzzy Systems, 21(5), 317-329.
Ozbakir, L., Tapkan, P., 2011. Bee colony intelligence in zone constrained two-sided assembly line balancing problem. Expert Systems with Applications, 38(9), 11947-11957.
Ozcan, U., 2010. Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. European Journal of Operational Research, 205(1), 81-97.

Ozcan, U., Cercioglu, H., Gokcen, H., Toklu, B., 2009. A Tabu Search Algorithm for the Parallel Assembly Line Balancing Problem. Gazi University Journal of Science, 22(4), 313-323.
Ozcan, U., Cercioglu, H., Gokcen, H., Toklu, B., 2010a. Balancing and sequencing of parallel mixed-model assembly lines. International Journal of Production Research, 48(17), 5089-5113.
Ozcan, U., Gokcen, H., Toklu, B., 2010b. Balancing parallel two-sided assembly lines. International Journal of Production Research, 48(16), 4767-4784.
Ozcan, U., Toklu, B., 2009. A tabu search algorithm for two-sided assembly line balancing. International Journal of Advanced Manufacturing Technology, 43(7-8), 822-829.
Ozcan, U., Toklu, B., 2010. Balancing two-sided assembly lines with sequence-dependent setup times. International Journal of Production Research, 48(18), 5363-5383.
Purnomo, H.D., Wee, H.M., Rau, H., 2013. Two-sided assembly lines balancing with assignment restrictions. Mathematical and Computer Modelling, 57(1-2), 189-199.
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.
Reed, M., Yiannakou, A., Evering, R., 2014. An ant colony algorithm for the multi-compartment vehicle routing problem. Applied Soft Computing, 15169-176.
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.
Scholl, A., Boysen, N., 2009. Designing parallel assembly lines with split workplaces: Model and optimization procedure. International Journal of Production Economics, 119(1), 90-100.
Shyu, S.J., Lin, B.M.T., Yin, P.Y., 2004. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. Computers & Industrial Engineering, 47(2-3), 181-193.
Simaria, A.S., Vilarinho, P.M., 2009. 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. Computers & Industrial Engineering, 56(2), 489-506.
Suer, G.A., 1998. Designing parallel assembly lines. Computers & Industrial Engineering, 35(3-4), 467-470.
Suer, G.A., Dagli, C., 1994. A knowledge-based system for selection of resource allocation rules and algorithms, in: Mital, A., Anand, S. (Eds.), Handbook of expert system applications in manufacturing; structures and rules. Chapman and Hall, London, pp. 108-147.
Taha, R.B., El-Kharbotly, A.K., Sadek, Y.M., Afia, N.H., 2011. A Genetic Algorithm for solving two-sided assembly line balancing problems. Ain Shams Engineering Journal, 2(3-4), 227-240.
Talbot, F.B., Patterson, J.H., 1984. An Integer Programming Algorithm with Network Cuts for Solving the Assembly Line Balancing Problem. Management Science, 30(1), 85-99.
Tavares Neto, R.F., Godinho Filho, M., 2013. Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research. Engineering Applications of Artificial Intelligence, 26(1), 150-161.
Tonge, F.M., 1960. Summary of a Heuristic Line Balancing Procedure. Management Science, 7(1), 21-42.
Tuncel, G., Aydin, D., 2014. Two-sided assembly line balancing using teaching-learning based optimization algorithm. Computers & Industrial Engineering.
Tuncel, G., Topaloglu, S., 2013. Assembly line balancing with positional constraints, task assignment restrictions and station paralleling: A case in an electronics company. Computers & Industrial Engineering, 64(2), 602-609.
Venkata Narasimha, K., Kivelevitch, E., Sharma, B., Kumar, M., 2013. An ant colony optimization technique for solving min–max Multi-Depot Vehicle Routing Problem. Swarm and Evolutionary Computation, 1363-73.
Vilarinho, P.M., Simaria, A.S., 2006. ANTBAL: an ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. International Journal of Production Research, 44(2), 291-303.

Wu, E.F., Jin, Y., Bao, J.S., Hu, X.F., 2008. A branch-and-bound algorithm for two-sided assembly line balancing. International Journal of Advanced Manufacturing Technology, 39(9-10), 1009-1015.
Yagmahan, B., 2011. Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. Expert Systems with Applications, 38(10), 12453-12461.
Yagmahan, B., Yenisey, M.M., 2008. Ant colony optimization for multi-objective flow shop scheduling problem. Computers & Industrial Engineering, 54(3), 411-420.
Yalcinkaya, O., Bayhan, G.M., 2009. Modelling and optimization of average travel time for a metro line by simulation and response surface methodology. European Journal of Operational Research, 196(1), 225-233.
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.
Yu, B., Yang, Z.Z., 2011. An ant colony optimization model: The period vehicle routing problem with time windows. Transportation Research Part E-Logistics and Transportation Review, 47(2), 166-181.
Yu, B., Yang, Z.Z., Yao, B.Z., 2009. An improved ant colony optimization for vehicle routing problem. European Journal of Operational Research, 196(1), 171-176.
Zhang, D.Z., Kucukkoc, I., 2013. Balancing Mixed-Model Parallel Two-Sided Assembly Lines, in: Amodeo, L., Dolgui, A., Yalaoui, F. (Eds.), Proceedings of the International Conference on Industrial Engineering and Systems Management (IEEE-IESM'2013), École Mohammadia d'Ingénieurs de Rabat (EMI), International Institute for Innovation, Industrial Engineering and Entrepreneurship (I4E2), Rabat, Morocco, pp. 391-401.

## Graphical Abstract


PTALBP-E
Problem Description & Mathematical Model
Operator 1
Operator 2
Operator 3
$L_1$
$L_2$
L
R
a
b
c
d
e
f
Good Quality Line Balancing Solutions
SOLUTION METHOD with ACO based Balancing Algorithm
Start
Calculate all parameters needed ($pt_{hmax}$, $C_{hLB}$, $C_{hUB}$, $C_{hInc}$, etc.)
$C_1 = C_{1LB}$; $C_2 = C_{2LB}$ $PBS^* = Very\ big\ number$
Find $LCM(C_1, C_2)$ and normalise task times
Build a balance for given $C_1$ and $C_2$ (use ACO algorithm)
$PBS < PBS^*$?
$PBS^* = PBS$
$C_2 = C_2 + C_{2Inc}$
$C_2 < C_{2UB}$
$C_1 < C_{1UB}$
$C_1 = C_1 + C_{1Inc}$ $C_2 = C_{2LB}$
Yes
No
Stop
Optimised ACO Algorithm Parameters
RESPONSE SURFACE METHODOLOGY to Optimise the ACO Algorithm Parameters
Optimal D: 1.000 Predict
High Cur Low
α 0.90 [0.7222] 0.10
β 0.90 [0.90] 0.10
ρ 0.650 [0.4621] 0.050
σ 30.0 [30.0] 10.0
Composite Desirability D: 1.000
BSE Maximum y = 94.2111 d = 1.0000
ASE Maximum y = 77.5649 d = 1.0000

**Research Highlights**

- Type-E parallel two-sided line balancing problem is introduced for the first time
- ACO algorithm is proposed as a possible solution approach for the addressed problem
- Parameters of the ACO are optimised through Response Surface Methodology
- The cycle time and the total number of workstations are minimised at the same time
- The performance of the ACO algorithm is tested through well-known test problems