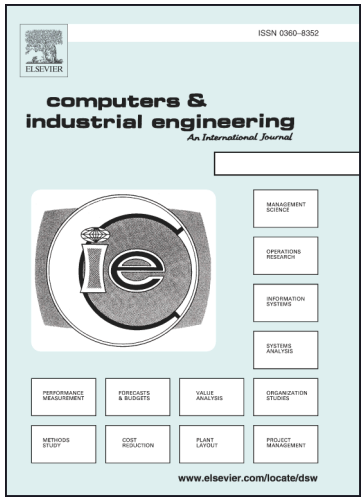
Accepted Manuscript



Mixed-model Parallel Two-sided Assembly Line Balancing Problem: A Flexible Agent-based Ant Colony Optimization Approach

Ibrahim Kucukkoc, David Z. Zhang

PII: S0360-8352(16)30101-2
DOI: http://dx.doi.org/10.1016/j.cie.2016.04.001
Reference: CAIE 4301

To appear in: *Computers & Industrial Engineering*

Received Date: 15 May 2015
Revised Date: 30 March 2016
Accepted Date: 1 April 2016

Please cite this article as: Kucukkoc, I., Zhang, D.Z., Mixed-model Parallel Two-sided Assembly Line Balancing Problem: A Flexible Agent-based Ant Colony Optimization Approach, *Computers & Industrial Engineering* (2016), doi: http://dx.doi.org/10.1016/j.cie.2016.04.001

Mixed-model Parallel Two-sided Assembly Line Balancing Problem: A Flexible Agent-based Ant Colony Optimization Approach

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

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

[b] Department of Industrial Engineering, Balikesir University, Cagis Campus, 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, Tel: +441392723613 (IK), +441392723641 (DZZ)

Abstract

Assembly lines are frequently used as a production method to assemble complex products. Two-sided assembly lines are utilized to assemble large-sized products (*e.g.*, cars, buses, trucks). Locating two lines in parallel helps improve line efficiency by enabling collaboration between the line workers. This paper proposes a mixed-model parallel two-sided assembly line system that can be utilized to produce large-sized items in an inter-mixed sequence. The mixed-model parallel two-sided line balancing problem is defined and the advantages of utilizing multi-line stations across the lines are discussed. A flexible agent-based ant colony optimization algorithm is developed to solve the problem and a numerical example is given to explain the method systematically. The proposed algorithm builds flexible balancing solutions suitable for any model sequence launched. The dynamically changing workloads of workstations (based on specific product models during the production process) are also explored. A comprehensive experimental study is conducted and the results are statistically analyzed using the well-known paired sample t-test. The test results indicate that the mixed-model parallel two-sided assembly line system reduces the workforce need in comparison with separately balanced mixed-model two-sided lines. It is also shown that the proposed algorithm outperforms the tabu search algorithm and six heuristics often used in the assembly line balancing domain.

Keywords: Assembly line balancing; Mixed-model lines, Parallel lines, Two-sided lines; Agent-based system; Ant colony optimization; Production planning.

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

1. Introduction

An assembly line consists of a sequence of workstations performing a limited set of repetitive tasks. Each task requires a certain amount of processing time and there are precedence relations between tasks (Battaïa and Dolgui, 2013). The main purpose of the assembly line balancing problem is to assign tasks to workstations in such a way that precedence relationships and capacity constraints (determined by demand and cycle time) are satisfied and some performance measures (*e.g.*, the number of workstations, line efficiency, *etc.*) are optimized. As the assembly lines are usually utilized towards the end of the production system, assembly line balancing has a significant effect on the overall performance of the entire manufacturing system, so it is one of the most significant challenges when designing and managing production lines (Kara *et al.*, 2010; Kucukkoc *et al.*, 2013).

It has been six decades since Salveson (1955) brought the assembly line balancing problem to the interest of academia and there has been continuing interest in this topic. Mixed-model lines emerged as a consequence of increasing interest in customized products in the global market, and Thomopoulos (1967) introduced the mixed-model assembly line balancing problem. Mixed-model lines enable production of different product models on the same line in an intermixed sequence (Boysen *et al.*, 2008; Kucukkoc and Zhang, 2014). Various heuristic, meta-heuristic, and exact solution approaches have been proposed for solving the mixed-model assembly line balancing problem; *e.g.*, Kara *et al.* (2007), Ozcan and Toklu (2009a), Ozcan *et al.* (2010a), Mosadegh *et al.* (2012), and Manavizadeh *et al.* (2013) developed simulated annealing approaches; Ozcan *et al.* (2011), Xu and Xiao (2011), Akpinar and Bayhan (2011), Hamzadayi and Yildiz (2012), and Manavizadeh *et al.* (2012) developed genetic algorithm techniques; Simaria and Vilarinho (2009) and Yagmahan (2011) developed ant colony optimization (ACO) techniques, and Akpinar *et al.* (2013) hybridized ACO with a genetic algorithm. Simaria and Vilarinho (2009) developed a mathematical model and Yagmahan (2011) considered multiple objectives as well as minimizing the total number of workstations.

Two-sided assembly lines, which are considered to be more practical for the assembly of large-sized products (*e.g.*, trucks) than of small ones (*e.g.*, electrical drills), were introduced by Bartholdi (1993). Different heuristics (*e.g.*, Lee *et al.* (2001), Hu *et al.* (2008), Ozcan and

Toklu (2010), and Yegul *et al*. (2010)) and meta-heuristics (*e.g.*, Kim *et al*. (2000; 2009), Baykasoglu and Dereli (2008), Simaria and Vilarinho (2009), Ozcan and Toklu (2009b), Ozcan (2010), Ozbakir and Tapkan (2010, 2011), Rabbani *et al*. (2012), Chutima and Chimklai (2012), and Purnomo *et al*. (2013)) were proposed for balancing the two-sided assembly lines. Some exact solution approaches have also been presented by Wu *et al*. (2008) and Hu *et al*. (2010) to solve the two-sided assembly line balancing problem optimally.

Recently, Gökçen *et al*. (2006) introduced a line parallelization idea, where two or more assembly lines with a common set of resources are balanced together, and defined the parallel assembly line balancing problem. They showed that locating two straight lines in parallel helps minimize the total number of workstations. This was followed by other researches and new techniques have been developed for solving the problem (*e.g.*, a shortest path approach by Benzer *et al.* (2007), a tabu search algorithm by Ozcan *et al.* (2009), an exact procedure by Scholl and Boysen (2009), a fuzzy goal programming model by Kara *et al.* (2010), and ACO based approaches by Baykasoglu *et al.* (2009) and Ozbakir *et al.* (2011)). The line parallelization idea has been applied to mixed-model lines and two-sided lines. Ozcan *et al*. (2010a) addressed parallel mixed-model lines and proposed a simulated annealing method for solving the balancing and sequencing problems on such lines. Ozcan *et al*. (2010b) introduced a parallel two-sided assembly line configuration to combine the advantages of both parallel lines and two-sided lines, and proposed a tabu search algorithm (referred to as TS hereafter). Kucukkoc and Zhang (2015) minimized cycle time and the number of workstations simultaneously on parallel two-sided lines using an ACO algorithm based approach.

As can be understood from this survey, there are numerous studies on mixed-model lines, two-sided lines, and parallel lines individually. There are a few studies on their binary combinations, *e.g.*, mixed-model parallel lines, parallel two-sided lines, *etc*. However, the number of studies on mixed-model parallel two-sided lines is fairly limited. Although mixed-model parallel two-sided assembly lines are encountered in producing large-sized high-volume products in industry (see for example Jack (2012) for Mercedes-Benz™ bus assembly lines at their Istanbul plant and Bloomberg (2015) for Toyota™ Caetano Portugal airport buses assembly line at their Portugal plant), this problem has not been properly dealt with by researchers. Kucukkoc and Zhang (2014) introduced the mixed-model parallel two-sided assembly line balancing and sequencing problem and proposed a solution framework to solve the problem using an agent-based approach.

This paper aims to develop a generic solution-building approach for the mixed-model assembly line balancing problem using the framework proposed by Kucukkoc and Zhang (2014) and contributes to knowledge in several aspects. First of all, unlike the study by Kucukkoc and Zhang (2014), the problem studied in the current work does not consider the model sequencing problem. Instead, it aims to obtain a mixed-model parallel two-sided assembly line system which is compatible with any model sequence. The solutions obtained thus are more flexible (have less criticalness) and can be used in a production environment where model demands are not predictable or changes happen in a very short time. Moreover, an agent based ant colony optimization (ABACO) algorithm is developed that will accommodate any model sequence and the performance of the proposed method is verified through experimental tests. Furthermore, a mathematical formula is developed to calculate the lower bound of line length and the number of workstations for mixed-model parallel two-sided lines. The results obtained through solving various test problems using the proposed approach are compared with the lower bounds calculated for these problems through statistical analysis. The same test problems are also solved using six well-known heuristics and a TS approach, and the results are compared with those results obtained from ABACO. See Appendices A.4 for a summary of differences between Kucukkoc and Zhang's study (2014) and the current article.

The remainder of this paper is organized as follows. Section 2 gives descriptive information on the mixed-model parallel two-sided assembly line balancing problem (referred to as MPTALBP hereafter). Section 3 explains the proposed solution method and Section 4 gives a numerical example. The results of computational experiments are exhibited in Section 5. Section 6 statistically analyzes the obtained results using the paired sample t-test and Section 7 presents the conclusion by emphasizing the limitations and possible industrial implications of the research as well as future work directions.

2. Problem definition

The MPTALBP is a new research domain and provides the flexibility to produce similar models of a large-sized product on parallel lines (Zhang and Kucukkoc, 2013). This new type of line configuration has many practical advantages such as a shorter line length, the shared use of common tools, flexibility to produce different product models at different throughput rates, lower material handling costs, fewer operator movements, and increased line efficiency with a lower number of workstations. Each two-sided line, which is located in parallel to its

adjacent two-sided line, is represented by L_h $(h = 1, \dots, H)$, and more than one different product model, symbolized by m_{hj} $(j = 1, \dots, M_h)$, is produced on each parallel two-sided assembly line. A set of tasks, t_{hji} $(i = 1, \dots, T_{hj})$, belonging to each product model is performed according to certain technological precedence relationships and P_{hji} represents the set of predecessors of task t_{hji} for model m_{hj} on line L_h. A certain amount of processing time (pt_{hji}) associated with each task needs to be processed on a series of workstations W_{hkx} $(k = 1, \dots, K_h;\ x = 0, 1)$, where x is a binary variable and 0 represents the left side line while 1 represents the right side line (Kucukkoc and Zhang, 2014). As in traditional lines, capacity constraints and precedence relationships need to be satisfied. However, special attention is required when assigning tasks which have precedence relationships between each other and are performed on different sides of the line. Assume that tasks 1 and 3 are predecessors of task 5, and tasks 3 and 5 can be performed on the left side of the line while task 1 requires the opposite side. In that case, task 1, which is assigned on the other side of the line, must be completed before initializing task 5. This situation is called *interference* in the line balancing literature and violation of this rule yields infeasible solutions.

An important advantage of parallel two-sided lines is the utilization of *multi-line stations*. Multi-line stations help minimize the total number of operators (or workstations) so that line efficiency is maximized. Figure 1 represents a typical mixed-model two-sided assembly line system.

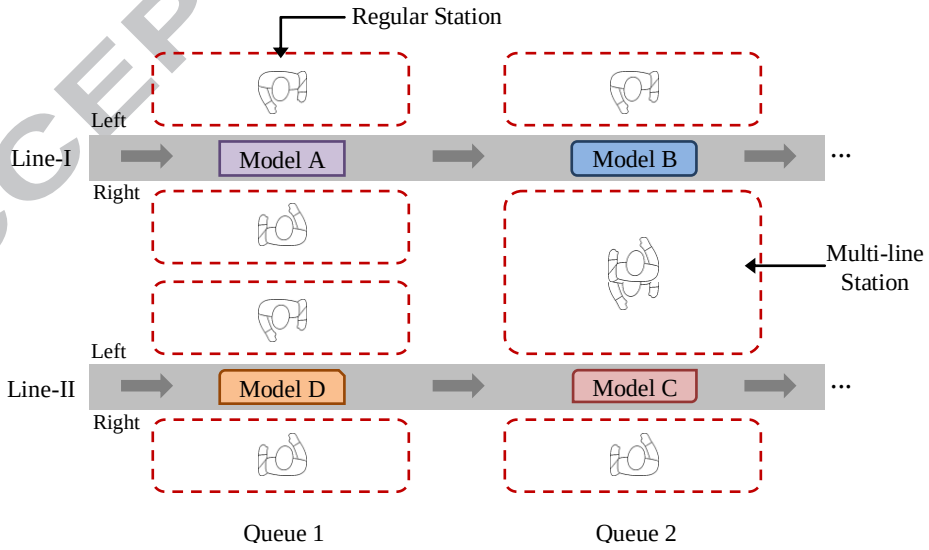


Figure 1. A typical mixed-model parallel two-sided assembly line system (Kucukkoc and Zhang, 2014)

As can be seen from the figure, the operator allocated at the multi-line station utilized between two adjacent lines in queue 2 works on both the right side of Line I and the left side of Line II. Each line may have a different cycle time (C_h):

$$C_h = \frac{P}{\sum_{j=1}^{M_h} D_{hj}}, \qquad h = 1, \dots, H, \tag{1}$$

where D_{hj} represents demand for model m_{hj} on line L_h over a planning period of P.

In case of cycle time differences, a common cycle time should be used to assign tasks in each cycle. This can be achieved using the least common multiple (LCM) based procedure used by Gökçen *et al*. (2006). In this procedure, a common cycle time (C) for two lines with different cycle times is calculated. The processing times of tasks on these lines are modified in accordance with the ratio of lines' original cycle time to common cycle time. To explain step-by-step (Gökçen *et al.,* 2006): (i) The LCM of the cycle times (C_1 and C_2) is found; (ii) LCM is divided by C_1 and C_2 to calculate ld_1 and ld_2, respectively; (iii) The processing times of tasks performed on Line I and Line II are multiplied by ld_1 and ld_2, respectively; (iv) LCM is considered as the common cycle time of the lines and the lines are balanced together.

The assumptions of the study are as follows:

- More than one product model is produced on each parallel two-sided assembly line.
- Task times and model demands are known and deterministic.
- Each line may have a different cycle time (determined by the demand over a planning horizon).
- Common tasks between similar models must be allocated to the same workstation to increase the resource utilization. For this aim, a joint precedence diagram is built for the product models produced on the same line.
- Tasks have preferences regarding the operation side, *i.e.,* left side, right side, or either side.
- Tasks cannot be split between workstations. Each task must be assigned to exactly one workstation.
- Operators do not have preferences for tasks and/or workstations.
- Parallel workstations are not allowed, so only one operator can work at a workstation.
- Operator travel times are ignored and no work in progress inventory is allowed.
- Breakdowns and/or failures are not considered and the assembly process is performed constantly during the planning horizon.

Based on the nature of two-sided assembly line balancing problems, more attention is needed to avoid interference which may occur due to any incomplete predecessor tasks on the other side of the line. Considering model variations on each of the lines (where multi-line stations

are allowed between the two adjacent lines) makes the problem, which is already an NP-hard class of combinatorial optimization problem, even more complex to solve. Therefore, determining the available times of operators located in multi-line stations between the two adjacent lines and identifying the model sequences that will be produced on the lines play a vital role in obtaining a feasible as well as a quality solution. As the processing time of a task may differ from one product model to another during the production process, the availability of an operator located in these workstations depends on the sequence of models assembled on the lines. However, this paper proposes a more generic solution which complies with any model sequence.

3. Proposed method

Metaheuristics are commonly used optimization techniques when solving complicated problems or large problem instances thanks to their ability to offer a desired trade-off between solution quality and computing time (Sörensen, 2015). Metaheuristics are also more flexible than exact methods as they are problem-independent solution algorithms which can be adapted to fit the needs of most real-life optimization problems; see, for example, Tiwari and Vidyarthi (2000), Tasan and Tunali (2008), and Pratap *et al.* (2015).

ACO is a nature-inspired metaheuristic algorithm which mimics the collective capability of real ant colonies to find the shortest path between the nest and a food source. Ants communicate with each other through a chemical substance called *pheromone*. At the beginning of the search process, ants randomly wander on the path. The ant which finds a source of food walks back to the colony, leaving pheromone on the ground. The path which includes pheromone on it will be followed by other ants at a certain probability. In this way, shorter paths are likely to be stronger because the ants will drop pheromone on the ground every time they walk back to the colony. A shortest path will eventually be constructed between the nest and the foraging area (Dorigo *et al.*, 1999). Refer to Dorigo and Stützle (2004) for a comprehensive review of the literature on various applications of ACO algorithms.

Agent-based systems benefit from a network of problem solvers in collaboration with each other to find solutions for complex problems that are beyond their individual capabilities. The algorithm developed in this study, ABACO, consists of an ACO-based line balancing procedure that is built over an agent-based platform. The platform has a structure which is composed of three layers: a facilitator agent (FA), a planning agent (PA), and a balancing

agent (BA). Agents existing in each layer interact with each other to build a complete balancing solution for the MPTALBP. Briefly, the overall process is coordinated by the FA while the PA provides data required by the BA to build a balancing solution. Unlike in the study by Kucukkoc and Zhang (2014), ABACO does not contain a model sequencing mechanism as the model sequencing problem is not considered. Instead, a more flexible line balance, which complies with any model sequence, is intended.

The pseudocode of ABACO is given in Figure 2. As seen from the figure, the FA initializes ABACO by reading the input data (*i.e.*, task processing times, precedence relationships, planning horizon, and model demands) from the spreadsheets and the parameters of the algorithm from the integrated user interface. These parameters include the number of colonies (*max_Colony*) and the number of ants in each colony (*colony_Size*). The precedence relationships data are transformed into a matrix so that the availability of tasks can be checked easily during the solution-building process at later stages.

Start ABACO

The FA reads and normalizes the input data (*i.e.*, task processing times, precedence relationships, model demands, and planning horizon)

The FA imports algorithm parameters for ACO, *i.e.*, the maximum number of colonies (*max_Colony*) and the number of ants in each colony (*colony_Size*).

The FA invokes the PA to calculate initial parameters

The PA determines the parameters required for the algorithm (cycle times of the lines, common cycle time, and normalized task processing times) and initializes algorithm parameters *(colony_nb=1, colony_Best=InfinitePositiveNumber, global_Best=InfinitePositiveNumber*)

While (*colony_nb < max_Colony*) **{**

The PA invokes the BA to build balancing solutions

The BA releases a new colony

Ant number is initialized (*ant_nb ← 1*)

While (*ant_nb < colony_Size*) **{**

A new ant is released and a heuristic is selected randomly for local search

The performance measure of the ant is initialized (*ant_Best=InfinitePositiveNumber*)

The ant builds a balancing solution and its quality (*ant_Best*) is calculated

The pheromone is released in accordance with the quality of the solution obtained (*ant_Best*)

If (*ant_Best < colony_Best*) **{**

colony_Best←ant_Best

} End if

ant_nb++

} End while

Results are returned to the PA

The PA calculates the final performance measures and sends to the FA

The FA compares the solutions with the global best solution
If (*colony_Best < global_Best*) **{**
global_Best←colony_Best
The FA invokes the BA to release double amount of pheromone on the path
} End if
colony_nb++
End while
The best solution (which has the performance measure of *global_Best*) is derived
Terminate

Figure 2. The pseudocode of ABACO

The PA is invoked by the FA to perform planning and computing parameter values. The PA uses the data imported by the FA to make initial calculations which are as follows:

- The cycle times of each line are determined based on the demand for models produced on each line and the planning horizon using Eq. (1) given in Section 2.
- A common cycle time between the lines is constituted and the task processing times are normalized in accordance with the LCM principle explained in Section 2.

An ACO algorithm based procedure is adopted by the BA to build solutions benefiting from the emerged success of collaborative ants in nature. Upon being invoked by the PA, the BA releases a new colony to build a balancing solution using the parameters and data structured by the PA. Each colony consists of a certain number of ants, determined by *colony_Size*, and each ant in the colony establishes a solution following the procedure, of which the pseudocode is depicted in Figure 3. For local search, the ACO algorithm is enhanced by the same ten commonly used heuristics as those used by Kucukkoc and Zhang (2014).

Start
Initialize all sets (*i.e.*, available tasks, and unassigned tasks) and parameters (*i.e.*, $st(k) \leftarrow 0$, *where* $st(k)$ *represents the station time of workstation k*)
Select a line randomly
While (there is one or more unassigned task) **{**
Select an operation side randomly
Determine all available tasks for the current position
If (there is at least one available task) **{**
Select a task (*i.e.*, task *i*) using pheromone level and heuristic information
Assign task *i* to the current available station and increase the station time: $st(k) \leftarrow st(k) + \max(t_{hji})$
Remove task *i* from the unassigned tasks list of the relevant line
} **else if** (there is no available task due to interference**) {**
Increase the station time of the current workstation: $st(k) \leftarrow st(\underline{k})$, where $\underline{k}$ is the

```
        companion of workstation k
    } else if (there is no available task due to insufficient capacity) {
        If (the current operation side lies between two lines and there is one or more available
        task from the other line) {
            Merge stations located on the left side of Line-I and the right side of Line-II
        } else {
            If (both sides of the current line reached full capacity) {
                Increase the station number (k++)
                Select the other line
            } else if (at least one side of the current line has not reached full capacity) {
                Alternate the operation side
            } End if
        } End if
    } End if
} End while
Terminate
```

Figure 3. The procedure used for building a balancing solution

When a new ant is appointed, it picks up a random heuristic and starts generating a balancing solution from any line or side randomly, unlike in the study by Kucukkoc and Zhang (2014). Available tasks are determined in accordance with the current position of the ant as it moves forward. An available task should satisfy the following constraints: *(i)* capacity (considering this task's longest processing time among all models), *(ii)* precedence relationships, and *(iii)* operation side. Among the available tasks, a task is selected in probability and assigned to the current position. When a new task is being selected, the selection probability of every available task (p_{ik}) is calculated using Eq. (2) and the one that has the highest selection probability value is selected by the ant.

$$p_{ik} = \frac{[\tau_{ik}]^{\alpha}[\eta_i]^{\beta}}{\sum_{y \epsilon Z_i}[\tau_{iy}]^{\alpha}[\eta_i]^{\beta}}, \quad (2)$$

where α and β are weighing parameters which determine the influence of pheromone and heuristic information in the task selection process, respectively. The indexes i and k indicate task and current workstation while Z_i is the list of candidate tasks when selecting task i. τ_{ik} is the pheromone amount existing between task i and workstation k, and η_i is the heuristic information of task i that comes from the randomly selected heuristic.A pheromone matrix, sized ixk, holds the pheromone amount between each task and each workstation. The ant moves forward by assigning available tasks one-by-one until all tasks are assigned. The pheromone is deposited between the tasks and workstations in which they are assigned. The

amount of pheromone that is deposited between task i and workstation k is calculated using Eq. (3) considering the objective function value (OBJ) of the solution obtained.

$$\tau_{ik} \leftarrow (1-\rho)\tau_{ik} + \Delta\tau_{ik}\,, \tag{3}$$

where $\Delta\tau_{ik} = Q/OBJ$; ρ and Q denote the evaporation rate and a user-determined parameter.

The formula presented in Eq. (4)-Eq. (6) aims to minimize a weighted summation of line length (LL) and the number of workstations (NS). Although the majority of the research in the literature considers NS as a unique performance index, the OBJ employed in ABACO considers LL as well as NS. The reason lying behind this idea is the fact that the length of a line may be a crucial decision factor, especially when construction space in the manufacturing plant is limited. The influences of LL and NS on the OBJ are controlled by weighting factors γ_1 and γ_2, respectively. For example, if the user thinks that LL has importance double that of the NS, he/she may simply set $\gamma_1 = 2$ and $\gamma_2 = 1$.

$$Min\ OBJ = \gamma_1 LL + \gamma_2 NS\,, \tag{4}$$

$$LL = max\{q_{hkx}\}, \qquad k = 1, \dots, K_h; \quad h = 1, \dots, H; \quad x \in \{0,1\}\,, \tag{5}$$

$$NS = \sum_{h=1}^{H} \sum_{k=1}^{K_h} \sum_{x\in\{0,1\}} U_{hkx}\,, \tag{6}$$

where q_{hkk} indicates the queue number in which workstation W_{hkx} is utilized and $max\{q_{hkx}\}$ corresponds to the length of the line system. U_{hkx} is a binary variable which equals 1 if workstation W_{hkx} is utilized on side x of line L_h, 0 otherwise.

After all ants in the colony complete their tour in such a way, the results are sent to the FA through the PA. If the best solution found by the ants in the colony is better than the global best solution found so far, the global best solution is updated and double the amount of pheromone is deposited on its edges to make this path favorable by other ants. A new colony is released by the BA and this cycle continues until the max_Colony number is exceeded, after which the global best solution is derived.

4. The lower bound formulation and a numerical example

This section presents the proposed lower bound calculation for mixed-model parallel two-sided lines and provides an example to explain the solution-building procedure of the ABACO approach developed in the previous section.

4.1. The lower bound calculation

The theoretical minimum NS (or lower bound) of a simple assembly line system is calculated by dividing the sum of the processing times of all tasks by the cycle time; this can be used to check the quality of a solution obtained for a problem when the optimal solution is not known. As ABACO applies a multi-objective balancing approach (*i.e.*, the objective considered by the proposed approach aims to minimize both LL and the NS), it is fair to compare each test case's OBJ value obtained by ABACO with the calculated lower bound of the OBJ for each test case $\left(LB_{TC}^{OBJ}\right)$. To recapitulate, each test case is constituted by a pair of test problems, *i.e.*, test problem-I and test problem-II. Therefore, the LB_{TC}^{OBJ} value of each test case is determined by the individual lower bounds of LL and NS of its test problem pairs (*i.e.*, $LB_{TP_I}^{LL}$ and $LB_{TP_I}^{NS}$ for test problem-I and $LB_{TP_{II}}^{LL}$ and $LB_{TP_{II}}^{NS}$ for test problem-II).

$$LB_{TC}^{OBJ} = \gamma_1 \cdot LB_{TC}^{LL} + \gamma_2 \cdot LB_{TC}^{NS}\,, \tag{7}$$

$$LB_{TC}^{LL} = \max_{pn\in\{I,II\}} \left\{LB_{TP_{pn}}^{LL}\right\}, \tag{8}$$

$$LB_{TC}^{NS} = \sum_{pn\in\{I,II\}} LB_{TP_{pn}}^{NS}\,, \tag{9}$$

where pn represents the corresponding test problem number. The first term in Eq. (7), LB_{TC}^{LL}, represents the length of the whole line system. The line which has the maximum LL defines the length of the whole parallel two-sided assembly line system (see Eq. (8)). However, the lower bounds of the NS of both lines (the second term in Eq. (7), $LB_{TP_{pn}}^{NS}$) are summed as in Eq. (9) to find the lower bounds for the total NS belonging to both lines.

To calculate the $LB_{TP_{pn}}^{NS}$ value of a test problem where a test problem is represented by pn, the formulation proposed by Simaria and Vilarinho (2009) for mixed-model two-sided assembly lines is used:

$$LB_{TP_{pn}}^{NS} = LB_{TP_{pn}L}^{NS} + LB_{TP_{pn}R}^{NS} + LB_{TP_{pn}E}^{NS}\,, \tag{10}$$

where $LB_{TP_{pn}L}^{NS}$ and $LB_{TP_{pn}R}^{NS}$ are the theoretical minimum number of left side and right side workstations for test problem pn and computed as in Eq. (11) and Eq. (12), respectively.

$$LB_{TP_{pn}L}^{NS} = \left\lceil \frac{\sum_{i\in S_L} \max_{j=1,\ldots,M_h} \{pt_{hji}\}}{C} \right\rceil^{+}, \tag{11}$$

$$LB_{TP_{pn}R}^{NS} = \left\lceil \frac{\sum_{i \in S_R} \max_{j=1,\dots,M_h} \{pt_{hji}\}}{C} \right\rceil^{+}, \quad (12)$$

where S_L and S_R are the sets of tasks that must be performed on the left side and right side of the line, respectively. C is the cycle time of the line that is subject to the relevant test problem while pt_{hji} corresponds to the processing time of task t_{hji}. As $LB_{TP_{pn}}^{NS}$ is calculated for each line individually, the 'h' subscript equals 1, which is given here to keep consistency with the parameters used in other sections of the study. The term $\lceil X \rceil^{+}$ denotes the least integer equal to or greater than X.

The term $LB_{TP_{pnE}}^{NS}$ is the theoretical minimum NS required to perform tasks that can be performed on either side (left side or right side) and is calculated as follows:

$$LB_{TP_{pnE}}^{NS} = \left\lceil \frac{\left[\sum_{i \in S_E} \max_{j=1,\dots,M_h} \{pt_{hji}\} - \left(\left(LB_{TP_{pnL}}^{NS} + LB_{TP_{pnR}}^{NS} \right) \cdot C - \sum_{i \notin S_E} \max_{j=1,\dots,M_h} \{pt_{hji}\} \right) \right]}{C} \right\rceil^{+}, \quad (13)$$

where S_E denotes the set of tasks which can be performed on either side of the line.

The formulation given above presents how $LB_{TP_{pn}}^{NS}$ is calculated for test problem pn. $LB_{TP_{pn}}^{LL}$ can be found using $LB_{TP_{pn}}^{NS}$. As the workstations can be established on two sides of the line, the LL of a two-sided line cannot be lower than half of the NS utilized on that line ($LL \geq NS/2$). Thus, given a lower bound for the NS, a lower bound of LL for a two-sided assembly line can be calculated as follows:

$$LB_{TP_{pn}}^{LL} = \left\lceil LB_{TP_{pn}}^{NS}/2 \right\rceil^{+}. \quad (14)$$

4.2. A numerical example

An example is given below to show the solution-building procedure used in the ABACO algorithm and to check its quality by means of lower bounds calculated based on the formula given in Section 4.1. For this task, two precedence relationship diagrams are taken from two well-known test problems, P16 (Lee *et al.*, 2001) and P24 (Kim *et al.*, 2000) for Line I and

Line II, respectively. Then, task times are generated between zero and the maximum task time of the original problem, as given in Table A1 (see Appendices).

As ABACO generates more flexible solutions regardless of the sequences of models being assembled on the lines, individual model demands are not important. However, the cumulative demands of individual models on each line determine the cycle time of each line, which are assumed to be 16 and 18 time-units/item for Line I and Line II, respectively. The LCM based approach (Gökçen et al., 2006) explained in Section 2 is used to maintain a common cycle time.

For this task, line divisors (ld_h) are calculated as $ld_1 = LCM(C_1, C_2)/C_1 = 144/16 = 9$ and $ld_2 = LCM(C_1, C_2)/C_2 = 144/18 = 8$. Then, the processing time of each task for models on Line I and Line II are multiplied by ld_1 and ld_2, respectively. $LCM(C_1, C_2) = 144$ is accepted as the common cycle time (C). The normalized task times (presented in Table A2) and common cycle time are used while balancing the lines.

An example ABACO solution-building procedure is shown in Figure 4. The balancing procedure starts from a randomly selected line and a randomly selected side (the right side of Line I in this example). Then, available tasks are determined and assigned to the workstations one-by-one using the procedure explained in the previous section. Arrows show the task assignment order for this example using ABACO.

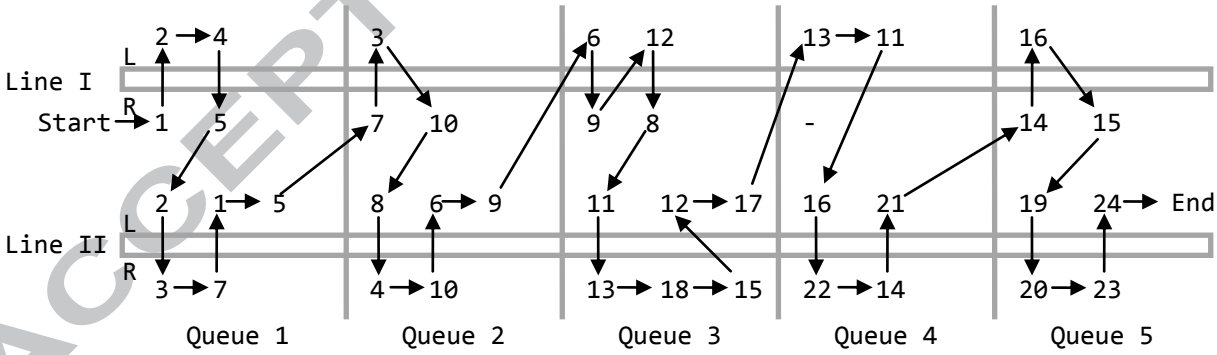


Figure 4. An example solution building procedure

Figure 4 presents the detailed balancing solution obtained for the numerical example problem. As seen from the figure, a total of 19 workstations are utilized across both of the lines where the length of the line system is 5 units. Thus, the OBJ value is simply calculated as 29 using Eqs. (4)-(6). Assignment configuration of tasks to the workstations can be clearly seen from Figure A1 (see Appendices), where the lengths of bars correspond to the processing times of tasks given in those bars. In this way, the changing workload of each workstation can be seen

for each particular model and line. As mentioned earlier, the maximum processing times of tasks among three models (given bold in Table A2) are considered to ensure that the capacity constraint is satisfied for any product model being assembled in the system. For example, if it is considered that the tasks are being assigned to the right side workstation with a remaining capacity of 63 time units on Line I in queue 2 and all predecessors of tasks 9 and 10 have already been assigned, only task 10 will be available to be assigned to the current position as the processing time of task 9 for model C (72 time units) exceeds the remaining capacity of the workstation, which is 63 time units.

The convergence of the OBJ, which is calculated using the total NS and LL, is depicted in Figure 5. In accordance with the weighting parameters used, the change in LL of the line affects the OBJ value as double of that of the total NS ($\gamma_1 = 2,\ \gamma_2 = 1$).

The lower bound of the OBJ can be calculated for this example problem using the formula provided in Section 4.1. The theoretical minimum NS is calculated as $LB_{TP_I}^{NS} = 8$ for Line I and $LB_{TP_{II}}^{NS} = 10$ for Line II, using Eqs. (10)-(13). Accordingly, the lower bounds of LL for Line I and Line II can also be calculated as $LB_{TP_I}^{LL} = 4$ and $LB_{TP_{II}}^{LL} = 5$, respectively, using Eq. (14). Finally, the theoretical lower bound of the OBJ for the problem is found as $LB_{TC}^{OBJ} = 28$ with the help of Eqs. (7)-(9).

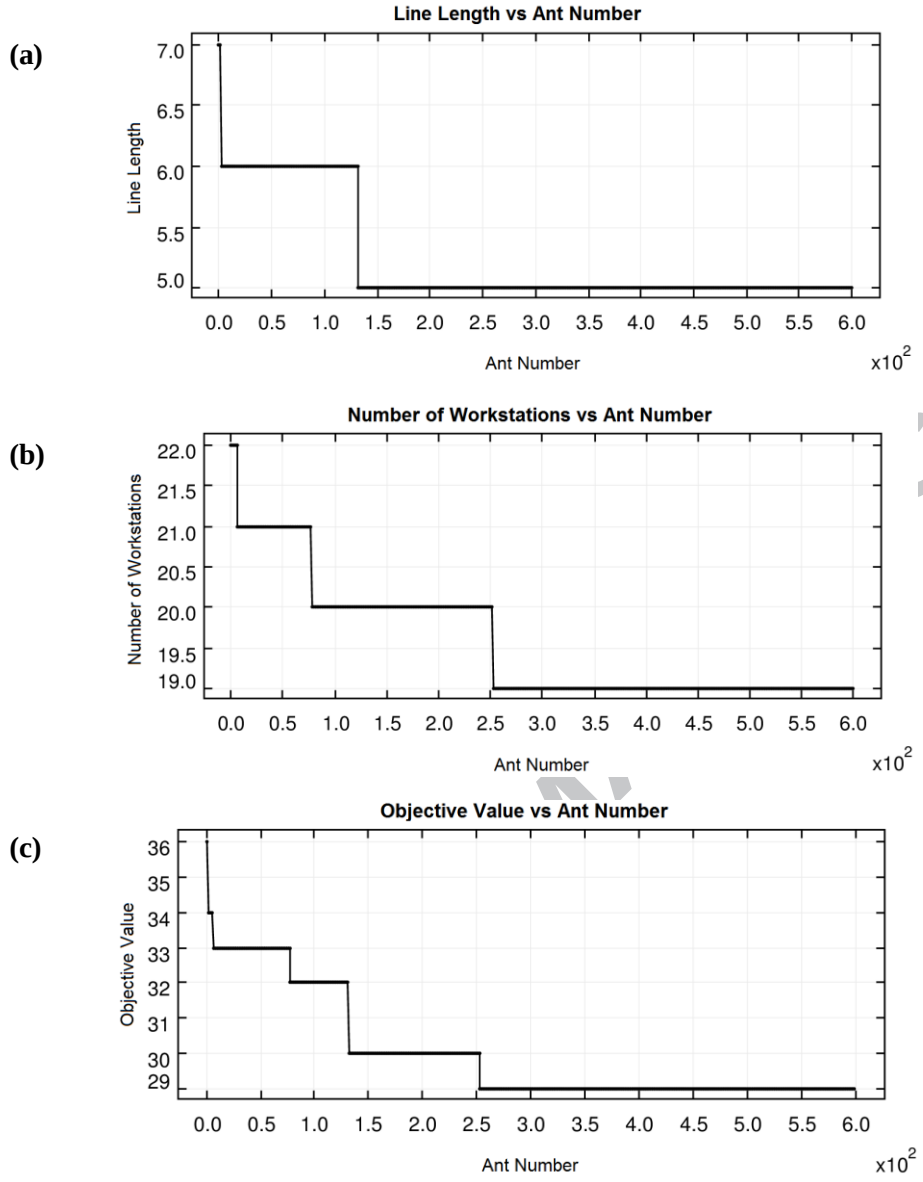


Figure 5. The convergence of (a) LL, (b) NS, and (c) OBJ

To recapitulate, the OBJ of the solution obtained by ABACO for this numerical example was 29 (see above). This provides us with information about the quality of this particular solution as the optimal solution of the same problem cannot be lower than 28 (whereas it does not also mean that the optimal OBJ value is 28). Therefore, it can be said that the solution found by ABACO for this example is near optimal (keeping in mind that there is a possibility of being optimal as well).

5. Computational experiments

5.1. Problem data

In order to analyze the efficiency of the proposed system and the performance of ABACO, 24 test cases were solved using ABACO and the obtained results were compared. Since there is no related study and so no published results in the literature with which to make comparisons, the same test cases were solved using six common heuristic approaches mentioned in Section 3: *(i)* a computer method of sequencing operations for assembly lines (COMSOAL), *(ii)* ranked positional weight method (RPWM), *(iii)* reverse ranked positional weight method (RRPWM), *(iv)* longest processing time (LPT), *(v)* least number of predecessors (LNP), and *(vi)* maximum number of successors (MNS). The results of ABACO were compared with those obtained from these six heuristics.

The original versions of these conventional heuristics address only the simple assembly line balancing problem, where only one model of a product is assembled on a one-sided line and no parallel lines are considered. Therefore, these techniques were adapted to solve the MPTALB/S problem and the same procedure was used with ABACO when determining available tasks.

The six heuristics and the proposed ABACO algorithm were coded in Java SE 7u4 and run on a 3.1 GHz Intel Core i5-2400 CPU computer. For the six heuristics, the algorithm was terminated after 100 iterations and the best solution was taken after one run for each test case. This means that each test case was solved 100 times by each heuristic and the final heuristic solution for each test case represents the best among the results obtained. For ABACO, parameters were chosen experimentally as shown in Table A3 to acquire a high-quality solution in a timely manner. As seen from the table, parameters differ based on the complexity of the problems. That is, increasing the capacity of ABACO as a search space enlarges with the increasing number of tasks.

To recapitulate, the same number of tasks were balanced in both separate balancing and together balancing conditions. In both approaches, two lines were balanced in the same run. Unlike separate balancing, the utilization of multi-line stations between two lines was allowed in the together balancing condition. This is why the same level of parameters has been applied in both situations.

Two parallel two-sided lines were assumed to be balanced with three models on each of the lines. Required data were generated using the seven original test problems from the literature; namely, P9, P12, and P24 from Kim *et al.* (2000); P16, A65, and A205 from Lee *et al.* (2001); and B148 from Bartholdi (1993). B148 was then modified by the work by Lee *et al.* (2001).

Also, the processing times of tasks 1, 131, 132, and 133 for problem A205 were modified to fit the rest of the data. Precedence relationship diagrams taken from these studies were considered as the common diagram for all three models on the same line, but new task times were generated randomly for the missing models. Table A4 provides the data used for test cases in this research.

5.2. Experimental results

Test cases with generated data were solved under various cycle time constraints and obtained results were compared with respect to *LL*, *NS*, and OBJ. Table 1 and Table 2 present the computational results when the lines were balanced separately and together, respectively. In together balancing, the utilization of multi-line stations was allowed between two lines.

As can be seen from Table 1, where the experimental results are reported for the separate balancing condition, ABACO found better solutions than all six of the test heuristics (namely COMSOAL, RPWM, RRPWM, LPT, LNP, and MNS) for 12 out of 24 test cases, *i.e.*, test cases 8, 11, 13, 14, 16–22, and 24 (see italicized *OBJ* values given in the ABACO column). For the remaining test cases, the results obtained by ABACO were either the same or better than some of the other approaches. The solution-building capacities of the heuristics get worse when the problem size gets bigger with the increasing number of tasks. Although no test heuristic found any better solution than ABACO for any of the test cases, MNS and RPWM found the same solutions as ABACO for eight and seven out of 24 test cases, respectively. For a better understanding, best results for each test problem are given in bold font in the table.

The solution-building capability of ABACO over six test heuristics improved in the together balancing condition, for which the experimental test results are presented in Table 2. ABACO performed better than the test heuristics for 14 test cases out of 24, *i.e.*, test cases 5, 6, 8, 9, 12, 13, 16, 17, and 19–24. To increase readability, ABACO results for these test cases are given in italic font in Table 2. Among six test heuristics, RPWM and MNS were the two heuristics which found closer results to those found by ABACO for the same test cases. However, all six of the test heuristics failed to investigate good-quality solutions when the problem size grew. Overall, these results indicate that the performance of ABACO was sufficient while balancing lines either separately or together. However, another significant outcome of the experimental tests performed in this section is that the OBJ was improved when the lines are balanced together. A statistical test is conducted in Section 6.1 for more comprehensive analysis on this.

Table 1. Computational results when the lines were balanced separately ($\gamma_1 = 2, \gamma_2 = 1$)

Test Case	COMSOAL			RPWM			RRPWM			LPT			LNP			MNS			ABACO		
	LL	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*
1	4	12	20	5	14	24	5	13	23	6	12	24	4	12	20	5	13	23	4	12	20
2	3	11	17	4	11	19	4	11	19	4	11	19	4	12	20	3	11	17	3	11	17
3	3	10	16	4	10	18	4	10	18	4	11	19	3	11	17	3	10	16	3	10	16
4	3	10	16	3	10	16	3	10	16	3	10	16	3	10	16	3	10	16	3	10	16
5	5	16	26	5	16	26	5	15	25	5	15	25	4	15	23	5	15	25	4	15	23
6	4	12	20	4	13	21	4	12	20	4	12	20	4	13	21	3	12	18	3	12	18
7	8	18	34	8	18	34	10	19	39	9	19	37	9	19	37	8	18	34	8	18	34
8	8	18	34	8	18	34	10	19	39	9	18	36	9	19	37	8	18	34	*8*	*17*	*33*
9	8	25	41	8	25	41	8	26	42	8	25	41	9	26	44	8	25	41	8	25	41
10	7	22	36	7	21	35	8	23	39	7	23	37	7	24	38	7	22	36	7	21	35
11	7	26	40	7	26	40	7	26	40	7	26	40	7	26	40	7	26	40	*7*	*25*	*39*
12	6	21	33	5	20	30	6	21	33	6	21	33	6	22	34	5	20	30	5	20	30
13	8	25	41	8	25	41	8	25	41	8	25	41	8	25	41	8	25	41	*7*	*24*	*38*
14	5	18	28	5	18	28	5	18	28	5	18	28	5	18	28	5	18	28	*5*	*17*	*27*
15	16	54	86	16	49	81	16	51	83	16	51	83	17	53	87	16	51	83	16	51	83
16	14	52	80	13	48	74	14	51	79	13	49	75	14	50	78	13	49	75	*13*	*47*	*73*
17	12	44	68	12	42	66	12	46	70	12	44	68	12	43	67	12	42	66	*11*	*41*	*63*
18	15	44	74	14	42	70	15	46	76	15	44	74	14	42	70	14	43	71	*14*	*41*	*69*
19	31	92	154	27	89	143	32	92	156	31	93	155	30	90	150	29	87	145	*26*	*86*	*142*
20	2	8	12	2	7	12	2	8	13	2	8	13	2	8	12	2	7	12	*2*	*7*	*11*

	3	2	8	1	7	1	3	5	1	3	4	0	3	1	7	2	9	3	*1*	*6*	*8*
21	25	81	131	25	78	128	27	85	139	27	85	139	26	81	133	25	80	130	*24*	*77*	*125*
22	23	71	117	21	67	109	23	74	120	22	71	115	22	69	113	22	69	113	*21*	*66*	*108*
23	23	89	135	21	83	125	24	92	140	23	90	136	23	87	133	22	87	131	21	83	125
24	27	82	136	26	79	131	27	87	141	27	84	138	26	81	133	26	79	131	*25*	*76*	*126*

Table 2. Computational results when the lines were balanced together ($\gamma_1 = 2, \gamma_2 = 1$)

Test Case	COMSOAL			RPWM			RRPWM			LPT			LNP			MNS			ABACO		
	LL	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*	*LL*	*NS*	*OBJ*
1	4	12	20	5	13	23	4	12	20	6	12	24	4	12	20	4	14	22	4	12	20
2	3	11	17	4	11	19	4	12	20	4	11	19	4	12	20	3	12	18	3	11	17
3	3	10	16	4	10	18	3	11	17	4	10	18	3	11	17	3	10	16	3	10	16
4	3	10	16	3	10	16	3	10	16	3	10	16	3	10	16	3	10	16	3	10	16
5	4	16	24	5	16	26	4	16	24	5	15	25	4	15	23	5	15	25	*4*	*14*	*22*
6	4	12	20	4	13	21	4	12	20	4	12	20	4	12	20	3	12	18	*3*	*11*	*17*
7	8	18	34	8	18	34	7	19	33	7	18	32	8	18	34	8	18	34	7	18	32
8	7	19	33	8	18	34	8	19	35	8	19	35	8	19	35	8	18	34	*7*	*18*	*32*
9	8	25	41	8	25	41	8	26	42	8	25	41	8	27	43	8	25	41	*7*	*25*	*39*
10	7	22	36	7	21	35	7	23	37	7	22	36	7	23	37	7	21	35	7	21	35
11	7	26	40	7	25	39	7	27	41	7	26	40	7	26	40	7	26	40	7	25	39
12	6	20	32	5	20	30	6	20	32	6	21	33	6	22	34	5	20	30	*5*	*19*	*29*
13	7	25	39	7	25	39	8	25	41	7	25	39	7	25	39	7	25	39	*7*	*24*	*38*

14	5	18	28	5	18	28	5	18	28	5	17	27	5	17	27	5	17	27	5	17	27
15	15	54	84	14	49	77	14	52	80	15	51	81	15	52	82	14	50	78	14	49	77
16	14	50	78	13	48	74	13	51	77	13	49	75	13	49	75	13	49	75	*13*	*46*	*72*
17	12	43	67	11	41	63	12	44	68	12	44	68	12	43	67	11	41	63	*11*	*40*	*62*
18	14	45	73	12	41	65	13	46	72	14	45	73	13	43	69	12	41	65	12	41	65
19	28	93	149	26	89	141	28	93	149	29	90	148	29	91	149	25	86	136	*24*	*85*	*133*
20	22	81	125	20	78	118	23	83	129	22	83	127	21	79	121	21	77	119	*20*	*77*	*117*
21	24	83	131	23	81	127	25	87	137	24	85	133	24	81	129	23	80	126	*22*	*79*	*123*
22	21	71	113	20	68	108	21	75	117	21	73	115	20	70	110	20	68	108	*19*	*67*	*105*
23	23	87	133	21	82	124	23	89	135	22	87	131	22	87	131	21	82	124	*21*	*81*	*123*
24	25	83	133	22	77	121	25	83	133	26	82	134	25	79	129	22	77	121	*22*	*76*	*120*

A further comparison is made here against a well-known metaheuristic, namely TS. A TS approach, a basic version of the algorithm developed by Ozcan *et al*. (2010b), was used for this task (see the Appendix for the steps of the TS used). The algorithm was run for 250, 750, and 1250 iterations for test cases 1–6, 7–14, and 15–24, respectively, to solve the same test cases solved by ABACO allowing multi-line stations (together balancing mode). The results are comparatively presented in Table 3.

As reported in Table 3, TS found one better solution than ABACO; see test case 13, for which the TS solution requires one less workstation than the solution found by ABACO for the same test case. On the other hand, ABACO found better solutions in terms of OBJ values for 10 test cases in comparison to TS (test cases 6, 12, 16, 17, and 19–24). For the remaining test cases, the OBJ values found by the two approaches are the same.

Table 3. The comparison of results obtained by TS and ABACO

TS	***Test case***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***
	LL	4	3	3	3	4	3	7	7	7	7	7	5
	NS	12	11	10	10	14	12	18	18	25	21	25	20
	OBJ	20	17	16	16	22	18	32	32	39	35	39	30
	Test case	***13***	***14***	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***23***	***24***
	LL	**7**	5	14	13	11	12	25	21	23	20	21	22

	NS	**23**	17	49	48	41	41	86	77	81	68	82	77
	OBJ	**37**	27	77	74	63	65	136	119	127	108	124	121
ABACO	***Test case***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***
	LL	4	3	3	3	4	**3**	7	7	7	7	7	**5**
	NS	12	11	10	10	14	**11**	18	18	25	21	25	**19**
	OBJ	20	17	16	16	22	**17**	32	32	39	35	39	**29**
	Test case	***13***	***14***	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***23***	***24***
	LL	7	5	14	**13**	**11**	12	**24**	**20**	**22**	**19**	**21**	**22**
	NS	24	17	49	**46**	**40**	41	**85**	**77**	**79**	**67**	**81**	**76**
	OBJ	38	27	77	**72**	**62**	65	**133**	**117**	**123**	**105**	**123**	**120**

6. Discussion and validation of results

This section first analyzes whether the proposed mixed-model parallel two-sided line system has a significant effect on minimizing LL and the NS in comparison to traditional separate balancing practice. This is followed by analyzing the performance of the proposed ABACO approach against theoretical lower bounds.

6.1. Analyzing the effect of balancing lines together on OBJ

A paired sample t-test is conducted here to test whether balancing lines together has a significant effect on minimizing the OBJ values obtained for the test cases. To do so, the results obtained by ABACO under two different conditions were analyzed: when the lines were balanced separately (ABACO Separate) and when the lines were balanced together (ABACO Together). The null and alternative hypotheses stated at the $a = 0.05$ significance level (95% confidence interval) for the means of OBJ values obtained from ABACO Together (μ_T) and ABACO Separate (μ_S) are as follows:

H_0: *There is no significant difference between the means of OBJ values obtained when the lines were balanced together and separately in favor of the alternative* ($\mu_T \geq \mu_S$).

H_1: *Balancing lines together significantly reduces the OBJ values in comparison to balancing lines separately* ($\mu_T < \mu_S$).

As seen from H_1, the test is left-tailed due to the smaller than condition '<' used. The results of the statistical test are reported in Table 4. As the calculated probability is much less than the considered significance level ($p = 0.000 \ll a = 0.05$), the null hypothesis of H_0 was rejected with very strong evidence. The calculated t_stat value also verified this result, $t\_stat = -3.781 \ll t\_critical = -1.714$. The statistical test results indicate that there was a significant difference in the OBJ mean values between balancing the lines together ($\mu_T =$

$57.33, SD_T = 41.00$) and balancing the lines separately ($\mu_S = 59.13, SD_S = 42.59$) for the solved test cases; $t(23) = -3.781, p = 0.000$. These results suggest that balancing mixed-model parallel two-sided assembly lines together has a significant effect on the OBJ values of the solutions obtained at $a = 0.05$ significance level. Thus, it is statistically shown that the OBJ values of obtained solutions are minimized when the mixed-model parallel two-sided lines are balanced together in comparison to balancing the lines separately.

Table 4. The results of paired sample t-test to analyze the effect of balancing lines together on the OBJ values of the solutions obtained

	ABACO Together	***ABACO Separate***
Mean (μ)	57.33	59.13
Standard deviation (SD)	41.00	42.59
Standard error mean	8.37	8.69
Variance	1680.7	1814.2
Observations	24	24
Degrees of freedom	23	-
Hypothesized mean difference	0	-
Pearson correlation	0.999	-
t_stat	**-3.781**	-
$p(T \leq t)$ **one-tail**	**0.000**	-
t_critical **one-tail**	**-1.714**	-

* The p-value is considered zero as it is smaller than 0.0001 ($p < 0.0001$).

6.2. The comparison of ABACO results against lower bounds

To validate the experimental test results, the results obtained by ABACO for 24 test cases under two conditions (when the lines were balanced separately and when the lines were balanced together) were compared against the lower bounds of these test cases. For this task, using the formula given in Section 4.1, the lower bound of the OBJ value for each test case was calculated and is presented in Table 5 in comparison with ABACO results. The $D(\%)$ column presents the deviation, which was calculated using Eq. (15), between the solution obtained by each procedure and the lower bound.

$$D(\%) = \left[\left(OBJ - LB_{TC}^{OBJ}\right)/LB_{TC}^{OBJ}\right] \cdot 100\,. \tag{15}$$

Table 5. Calculated lower bounds for the solved test cases ($\gamma_1 = 2, \gamma_2 = 1$)

Test Case	**Calculated Lower Bounds**								**ABACO Separate**				**ABACO Together**			
	$LB_{TP_I}^{LL}$	$LB_{TP_I}^{LL}$	$LB_{TP_{II}}^{LL}$	$LB_{TP_{II}}^{LL}$	LB_{TC}^{LL}	LB_{TC}^{NS}	LB_{TC}^{OBJ}		*LI*	*NS*	*OBI*	*D(%)*	*LI*	*NS*	*OBI*	*D(%)*
1	4	7	2	4	4	11	19		4	12	20	5.26	4	12	20	5.26
2	3	5	3	6	3	11	17		3	11	17	0.00	3	11	17	0.00
3	3	6	2	4	3	10	16		3	10	16	0.00	3	10	16	0.00

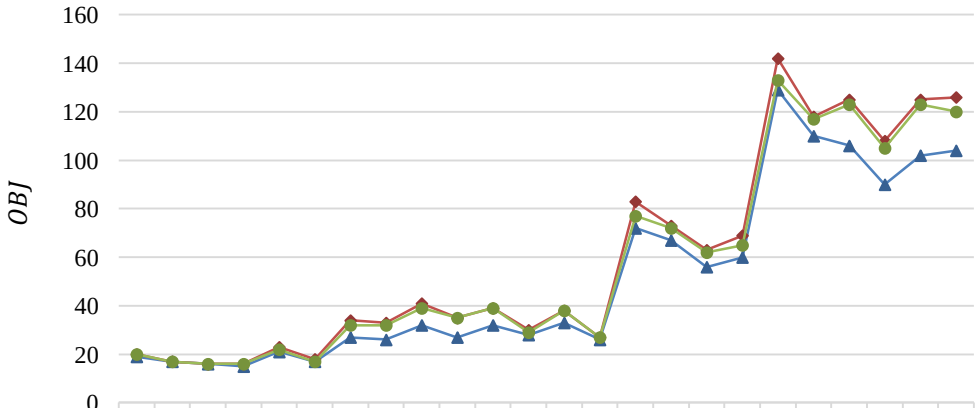
4	2	4	3	5	3	9	15		3	10	16	6.67	3	10	16	6.67
5	4	7	3	6	4	13	21		4	15	23	9.52	4	14	22	4.76
6	3	5	3	6	3	11	17		3	12	18	5.88	3	11	17	0.00
7	2	4	6	11	6	15	27		8	18	34	25.9	7	18	32	18.5
8	2	3	6	11	6	14	26		8	17	33	26.9	7	18	32	23.0
9	6	11	5	9	6	20	32		8	25	41	28.1	7	25	39	21.8
10	4	8	5	9	5	17	27		7	21	35	29.6	7	21	35	29.6
11	5	9	6	11	6	20	32		7	25	39	21.8	7	25	39	21.8
12	4	8	5	10	5	18	28		5	20	30	7.14	5	19	29	3.57
13	6	12	5	9	6	21	33		7	24	38	15.1	7	24	38	15.1
14	4	7	5	9	5	16	26		5	17	27	3.85	5	17	27	3.85
15	14	27	9	17	14	44	72		1	51	83	15.2	1	49	77	6.94
16	10	20	12	23	12	43	67		1	47	73	8.96	1	46	72	7.46
17	10	20	8	16	10	36	56		1	41	63	12.5	1	40	62	10.7
18	6	12	12	24	12	36	60		1	41	69	15.0	1	41	65	8.33
19	26	51	13	26	26	77	129		2	86	142	10.0	2	85	133	3.10
20	16	31	20	39	20	70	110		2	76	118	7.27	2	77	117	6.36
21	13	26	20	40	20	66	106		2	77	125	17.9	2	79	123	16.0
22	11	22	17	34	17	56	90		2	66	108	20.0	1	67	105	16.6
23	17	34	17	34	17	68	102		2	83	125	22.5	2	81	123	20.5
24	21	41	11	21	21	62	104		2	76	126	21.1	2	76	120	15.3
						Mean (μ)						14.0				11.0
						Standard Deviation (σ)						8.95				8.47

As can be seen from the comparison given in Table 5, the smallest difference (0.00%) between the lower bound and the OBJ values was observed for test cases 2 and 3 when the lines were balanced separately; and test cases 2, 3, and 6 when the lines were balanced together. This means that ABACO found optimal solutions for at least two test cases when the lines were balanced separately and three test cases when the lines were balanced together. For 16 test cases out of 24, the deviation between the lower bound and ABACO Together results was smaller than the deviation between the lower bound and ABACO Separate results (test cases 5-9, 12, and 15-24). For the remaining eight test cases, OBJ values obtained in both conditions have the same deviation from the lower bounds (test cases 1-4, 10, 11, 13, and 14).

The largest difference between the lower bound and the OBJ values found by ABACO in both separate balancing (given in the ABACO Separate column) and together balancing (given in the ABACO Together column) conditions was 29.63% (test case 10). However, it can be said that the algorithms perform fairly well in general as the calculation of the lower bound does not consider precedence relationship constraints. Furthermore, it is not always possible to fill up every workstation to full capacity as the processing times of tasks do not distribute homogeneously. This is supported by the unavoidable idle times, which cause an increase in the total NS. Furthermore, the two-sided and mixed-model nature of the considered system

reinforces this conclusion because unavoidable idle time occurs when a different model comes to the system which requires less time for a specific task than the previous model.

The deviation of the results given in the ABACO Together column from the lower bound ($\mu = 11.08, \sigma = 8.47$) was relatively less (better) than the deviation of the results given in the ABACO Separate column from the lower bound ($\mu = 14.03, \sigma = 8.95$). This further evidences the advantage of balancing lines together and utilizing multi-line stations between the two adjacent lines.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Lower Bound	19	17	16	15	21	17	27	26	32	27	32	28	33	26	72	67	56	60	129	110	106	90	102	104
Separate	20	17	16	16	23	18	34	33	41	35	39	30	38	27	83	73	63	69	142	118	125	108	125	126
Together	20	17	16	16	22	17	32	32	39	35	39	29	38	27	77	72	62	65	133	117	123	105	123	120

Figure 6. The graphical demonstration of obtained results with lower bounds

The best results obtained by ABACO for separate balancing and together balancing conditions are plotted in Figure 6 in comparison with calculated lower bound values. As can be seen from the figure, the gap between the lower bound and the obtained results opens slightly in large-sized instances: see test cases 21-24. As known, this is due to the increased complexity of large-sized test cases as the search space grows dramatically when the number of tasks increases. To recapitulate, the utilization of multi-line stations is allowed when the lines are balanced together, which is one of the essential advantages of such systems. It is obvious that this feature helps minimize the OBJ for test cases 5-9, 12, and 15-24.

7. Conclusions

The MPTALBP, which is an NP-hard class of combinatorial problem, has been defined. A sophisticated solution algorithm, which combines the ant colony optimization algorithm with 10 heuristics in an agent-based environment, has been developed for solving the generic balancing problem. The developed solution technique accommodates any model sequence for

products. Ants in the colony were allowed to select a random behavior among the provided heuristics so the local search capability of ABACO has been increased.

To show the superiority of the proposed approach, 24 test cases have been conducted under two different conditions: the lines were balanced together (*i.e.*, multi-line stations were allowed) and the lines were balanced individually (*i.e.*, multi-line stations were not allowed) and research results have been reported. The results have been compared and a paired sample t-test has been conducted to prove statistically whether balancing lines together helps minimize the performance measure, which is a weighted combination of LL and the NS, in comparison to balancing lines individually. The statistical test results indicate that balancing lines together reduces the LL and the NS. Comparative experiments have also been performed by solving the same 24 test problems using six well-known heuristics and a TS approach.

A lower bound formulation of the problem has also been developed building on work from the literature, and lower bound values for the objectives have been calculated for test cases used in this research. The results obtained from the proposed algorithm for each test case have been compared to the lower bounds calculated, and a comprehensive analysis has been presented. Deviation from the lower bounds regarding the obtained OBJ values has also been compared and analyzed.

The limitation of the work is that as the model sequencing problem is not considered along with the line balancing problem, obtained OBJ values are not as good as desired although the line is more flexible to demand changes and any new model sequences could be launched with no need for balance change. As future research directions, some new approaches could be applied to overcome this issue and two conflicting objectives, namely minimizing cycle time and the total number of required workstations, could be handled concurrently as an ultimate goal. A mathematical model can also be developed and optimal solutions can be compared with the results reported in this research. However, this may not be possible for medium or large-sized problems as it is hard to obtain optimal solutions when the problem size increases.

Acknowledgment

The authors acknowledge the valuable comments and suggestions of anonymous referees and the editorial team of Computers & Industrial Engineering which helped improve the presentation and quality of the manuscript.

Appendices

A.1. Pseudocode of TS

The pseudocode of the TS approach used in the paper is presented below. Please note that this is a simplified version of the well-known TS algorithm. This is because it is very hard to create and evaluate all candidate moves due to the sophisticated line balancing procedure developed and stochastic decisions allowed during the balancing process.

Start

Import and normalize input data.

Initialize all parameters, $iterationNumber \leftarrow 1$, $tabuList_I \leftarrow \{\}$, $tabuList_{II} \leftarrow \{\}$.

Create random priority indexes for tasks belonging to Line I and Line II, $priorityIndexes_I$ and $priorityIndexes_{II}$, respectively. For example, when creating $priorityIndexes_I$, assign each task a value between 1 and the total number of tasks on Line I, where each task gets a different priority index value. Apply the same procedure for $priorityIndexes_{II}$.

While ($\boldsymbol{iterationNumber} < maxNumberOfIterations$)

Build a balancing solution using the procedure given in Figure 3 but use priority index values of tasks instead of pheromone level and heuristic information when selecting tasks. For example if there are three available tasks (one from Line I and two from Line II) for the current position, the task which has the lowest priority index will be selected and assigned to this position, no matter the task is from which line.

Calculate the performance measure of the solution using Eqs. (4)-(6).

Update the current best solution if a better performance measure is observed.

Generate four random integers (r_1, r_2, r_3, and r_4) ; *i.e.*, r_1 and r_2 between 1 and $card\{priorityIndexes_I\}$, r_3 and r_4 between 1 and $card\{priorityIndexes_{II}\}$, where $card\{X\}$ denotes the length of list X.

If ($\{\boldsymbol{r_1, r_2}\} \notin \boldsymbol{tabuList_I}$)

Swap priority indexes of tasks r_1 and r_2 in the list of $priorityIndexes_I$.

Add $\{r_1, r_2\}$ to $tabuList_I$.

Else if ($\{\boldsymbol{r_1, r_2}\} \in \boldsymbol{tabuList_I}$)

Generate new r_1 and r_2 until $\{r_1, r_2\} \notin tabuList_I$.

End if

If ($\{\boldsymbol{r_3, r_4}\} \notin \boldsymbol{tabuList_{II}}$)

Swap priority indexes of tasks r_3 and r_4 in the list of $priorityIndexes_{II}$.

Add $\{r_3, r_4\}$ to $tabuList_{II}$.

Else if ($\{\boldsymbol{r_3, r_4}\} \in \boldsymbol{tabuList_{II}}$)

Generate new r_3 and r_4 until $\{r_3, r_4\} \notin tabuList_{II}$.

End if

If ($\boldsymbol{tabuList_I \geq maxTabuSize_I}$)

Remove first element in $tabuList_I$.

End if

If ($tabuList_{II} \geq maxTabuSize_{II}$)

Remove first element in $tabuList_{II}$.

End if

$iterationNumber \leftarrow iterationNumber + 1$.

End while

Report the current best solution.

Terminate

A.2. Data Used for the Numerical Example and Computational Tests

Table A1. Data for the numerical example

Line I						Line II					
Task no	Side	A	B	C	Immediate predecessors	Task no	Side	D	E	F	Immediate predecessors
1	E	6	7	6	-	1	L	3	3	0	-
2	E	5	2	0	-	2	L	7	0	2	-
3	L	2	5	9	1	3	R	7	1	1	-
4	E	9	2	8	1	4	R	5	0	0	-
5	R	8	9	5	2	5	L	4	6	1	2
6	L	4	8	0	3	6	E	3	5	1	2, 3
7	E	7	8	9	4, 5	7	R	4	8	5	3
8	E	4	6	3	6, 7	8	E	3	0	7	5
9	R	5	0	8	7	9	E	6	4	4	6
10	R	4	4	7	7	10	E	4	2	9	7
11	E	6	5	7	8	11	L	4	8	3	1
12	L	5	6	6	9	12	L	3	1	1	8, 9
13	E	6	4	9	9, 10	13	E	3	5	3	9
14	E	4	2	7	11	14	R	9	4	3	9, 10
15	E	3	6	9	11, 12	15	R	5	1	4	4
16	E	4	8	8	13	16	L	9	1	2	11
-	-	-	-	-	-	17	E	2	7	3	12
-	-	-	-	-	-	18	E	7	4	4	13
-	-	-	-	-	-	19	E	9	2	1	13, 14
-	-	-	-	-	-	20	R	9	1	1	15
-	-	-	-	-	-	21	L	8	9	7	16, 17
-	-	-	-	-	-	22	E	8	7	9	18
-	-	-	-	-	-	23	R	9	9	5	19, 20
-	-	-	-	-	-	24	E	9	3	5	20

Table A2. Data with normalized task processing times

Line I						Line II					
Task no	Side	A	B	C	Immediate predecessors	Task no	Side	A	B	C	Immediate predecessors
1	E	54	**63**	54	-	1	L	**24**	**24**	0	-
2	E	**45**	18	0	-	2	L	**56**	0	16	-

3	L	18	45	**81**	1	3	R	**56**	8	8	-
4	E	**81**	18	72	1	4	R	**40**	0	0	-
5	R	72	**81**	45	2	5	L	32	**48**	8	2
6	L	36	**72**	0	3	6	E	24	**40**	8	2, 3
7	E	63	72	**81**	4, 5	7	R	32	**64**	40	3
8	E	36	**54**	27	6, 7	8	E	24	0	**56**	5
9	R	45	0	**72**	7	9	E	**48**	32	32	6
10	R	36	36	**63**	7	10	E	32	16	**72**	7
11	E	54	45	**63**	8	11	L	32	**64**	24	1
12	L	45	**54**	**54**	9	12	L	**24**	8	8	8, 9
13	E	54	36	**81**	9, 10	13	E	24	**40**	24	9
14	E	36	18	**63**	11	14	R	**72**	32	24	9, 10
15	E	27	54	**81**	11, 12	15	R	**40**	8	32	4
16	E	36	**72**	**72**	13	16	L	**72**	8	16	11
-	-	-	-	-	-	17	E	16	**56**	24	12
-	-	-	-	-	-	18	E	**56**	32	32	13
-	-	-	-	-	-	19	E	**72**	16	8	13, 14
-	-	-	-	-	-	20	R	**72**	8	8	15
-	-	-	-	-	-	21	L	64	**72**	56	16, 17
-	-	-	-	-	-	22	E	64	56	**72**	18
-	-	-	-	-	-	23	R	**72**	**72**	40	19, 20
-	-	-	-	-	-	24	E	**72**	24	40	20

Table A3. The parameters of ABACO

Problem no	**Alpha**	**Beta**	**Evaporation rate**	**Initial Pheromone**	**Colony size**	**Number of Iterations**
1-6	0.1	0.2	0.1	10	10	10
7-14	0.1	0.2	0.1	15	20	30
15-24	0.1	0.2	0.1	20	30	60

Table A4. Data for test cases

Problem Scale	**Test Case**	**Problem**		**Cycle Time**		**Minimum Part Set (MPS)**					
		Line I	***Line II***	***Line I***	***Line II***	***Line I***				***Line II***	
						A	***B***	***C***		***D***	***E***	***F***
Small	1	P9	P9	4	7	4	2	1		2	1	1
	2	P9	P9	6	5	2	2	1		1	2	1
	3	P9	P12	5	8	2	1	1		2	2	1
	4	P9	P12	7	6	1	1	2		2	1	4
	5	P12	P12	4	5	2	1	2		1	2	1
	6	P12	P12	6	5	2	1	2		2	1	1
Medium	7	P12	P16	9	12	1	2	1		1	1	1
	8	P12	P16	10	12	1	1	1		1	2	2
	9	P16	P16	12	15	1	2	2		2	1	1
	10	P16	P16	16	14	1	4	2		2	1	1
	11	P16	P24	14	16	2	1	1		4	2	1
	12	P16	P24	16	18	1	3	2		1	2	1
	13	P24	P24	15	20	2	1	1		1	1	1
	14	P24	P24	25	20	1	2	1		1	2	2
Large	15	A65	A65	300	480	2	1	1		2	1	2
	16	A65	A65	420	360	1	1	2		2	4	1

	17	A65	B148	405	810	2	1	1		2	2	1
	18	A65	B148	675	540	2	1	1		1	2	2
	19	B148	B148	255	510	1	2	1		1	2	2
	20	B148	B148	425	340	2	1	1		2	2	1
	21	B148	A205	510	1020	2	1	3		1	2	2
	22	B148	A205	600	1200	2	1	1		1	1	1
	23	A205	A205	1200	1200	1	1	1		1	1	1
	24	A205	A205	1000	2000	3	1	2		1	1	1

A.3. Detailed Solution of the Numerical Example

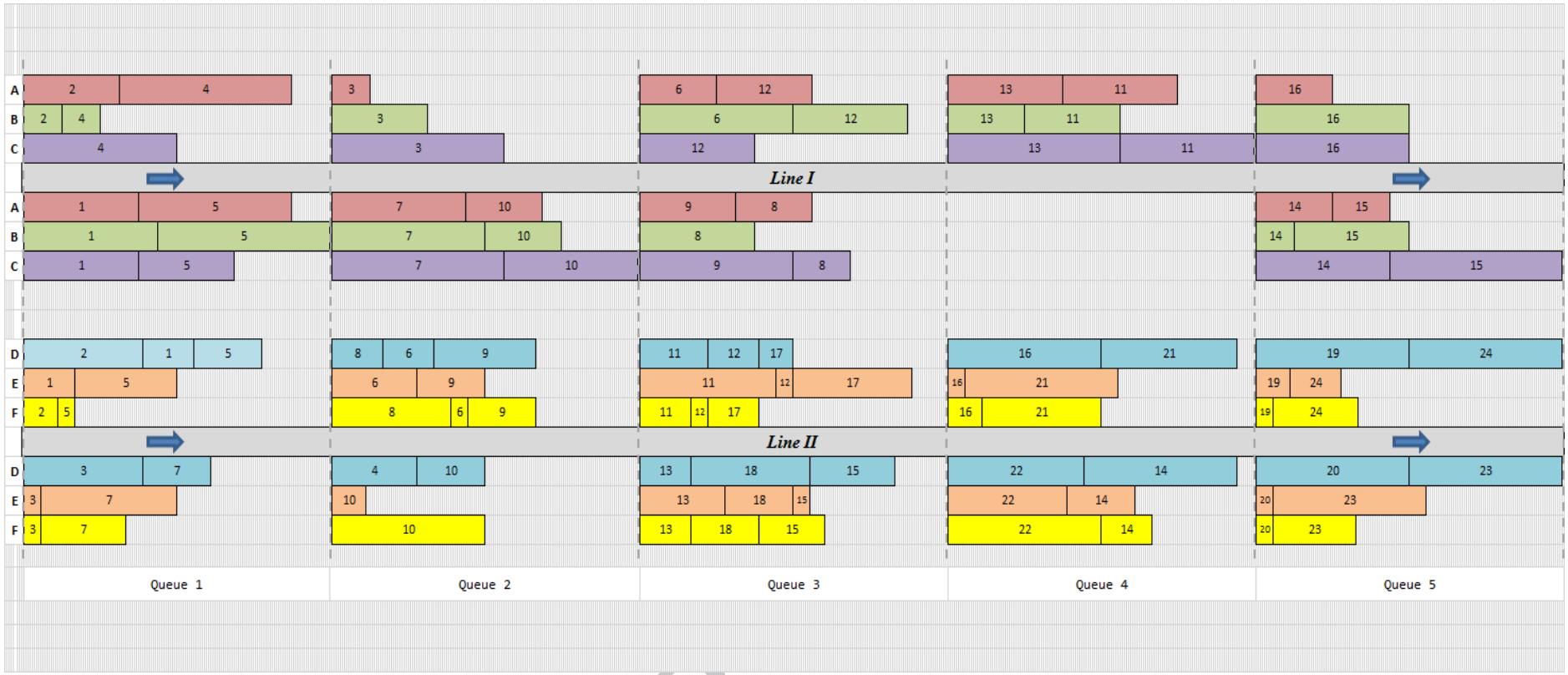


Figure A1. Detailed balancing solution and assignment configuration of tasks

A.4. Contribution summary

Table A5. Contributions over Kucukkoc and Zhang (2014)

	Results presented	Adaptability to demand changes	Model sequencing needed?	Solution complexity
Kucukkoc and Zhang (2014)	No	Low	Yes	High
Current article	Yes	High	No	Low

References

Akpinar, S., Bayhan, G.M., 2011. A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Engineering Applications of Artificial Intelligence 24, 449-457.

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, 574-589.

Bartholdi, J.J., 1993. Balancing 2-Sided Assembly Lines - a Case-Study. International Journal of Production Research 31, 2447-2461.

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

Baykasoglu, A., Dereli, T., 2008. Two-sided assembly line balancing using an ant-colony-based heuristic. International Journal of Advanced Manufacturing Technology 36, 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 1-3, 506-511.

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.

Bloomberg, 2015. A view from Toyota Caetano Portugal S.A. airport buses assembly line in Portugal plant, http://www.gettyimages.co.uk/detail/news-photo/contrac-cobus-3000-airport-buses-stand-on-the-production-news-photo/458163961, Accessed February 14, 2015.

Boysen, N., Fliedner, M., Scholl, A., 2008. Assembly line balancing: Which model to use when? International Journal of Production Economics 111, 509-528.

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, 39-55.

Dorigo, M., Di Caro, G., Gambardella, L.M., 1999. Ant Algorithms for Discrete Optimization. Artificial Life 5, 137–172.

Dorigo, M., Stutzle, T., 2004. Ant Colony Optimization. Bradford Books, MIT Press, Cambridge, MA.

Gökçen, H., Agpak, K., Benzer, R., 2006. Balancing of parallel assembly lines. International Journal of Production Economics 103, 600-609.

Hamzadayi, A., Yildiz, G., 2012. A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints. Computers & Industrial Engineering 62, 206-215.

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, 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, 435-440.

Jack, D., 2012. A view from Mercedes-Benz bus assembly plant in Istanbul, http://busride.com/wp-content/uploads/2012/05/web-LFE-turkey1.jpg, Accessed February 14, 2015.

Kara, Y., Gokcen, H., Atasagun, Y., 2010. Balancing parallel assembly lines with precise and fuzzy goals. International Journal of Production Research 48, 1685-1703.

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

Kim, Y.K., Kim, Y.H., Kim, Y.J., 2000. Two-sided assembly line balancing: a genetic algorithm approach. Production Planning & Control 11, 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, 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, 5039-5054, doi: http://dx.doi.org/5010.1080/00207543.00202013.00784411.

Kucukkoc, I., Zhang, D.Z., 2014. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Research 52, 3665-3687, doi: http://dx.doi.org/3610.1080/00207543.00202013.00879618.

Kucukkoc, I., Zhang, D.Z., 2015. Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. Computers & Industrial Engineering 84, 56-69.

Lee, T.O., Kim, Y., Kim, Y.K., 2001. Two-sided assembly line balancing to maximize work relatedness and slackness. Computers & Industrial Engineering 40, 273-292.

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

Manavizadeh, N., Rabbani, M., Moshtaghi, D., Jolai, F., 2012. Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms. Expert Systems with Applications 39, 12026-12031.

Mosadegh, H., Zandieh, M., Ghomi, S.M.T.F., 2012. Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines. Applied Soft Computing 12, 1359-1370.

Ozbakir, L., Baykasoglu, A., Gorkemli, B., Gorkemli, L., 2011. Multiple-colony ant algorithm for parallel assembly line balancing problem. Applied Soft Computing 11, 3186-3198.

Ozbakir, L., Tapkan, P., 2010. Balancing fuzzy multi-objective two-sided assembly lines via Bees Algorithm. Journal of Intelligent & Fuzzy Systems 21, 317-329.

Ozbakir, L., Tapkan, P., 2011. Bee colony intelligence in zone constrained two-sided assembly line balancing problem. Expert Systems with Applications 38, 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, 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, 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, 5089-5113.

Ozcan, U., Gokcen, H., Toklu, B., 2010b. Balancing parallel two-sided assembly lines. International Journal of Production Research 48, 4767-4784.

Ozcan, U., Kellegoz, T., Toklu, B., 2011. A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. International Journal of Production Research 49, 1605-1626.

Ozcan, U., Toklu, B., 2009a. Balancing of mixed-model two-sided assembly lines. Computers & Industrial Engineering 57, 217-227.

Ozcan, U., Toklu, B., 2009b. A tabu search algorithm for two-sided assembly line balancing. International Journal of Advanced Manufacturing Technology 43, 822-829.

Ozcan, U., Toklu, B., 2010. Balancing two-sided assembly lines with sequence-dependent setup times. International Journal of Production Research 48, 5363-5383.

Pratap, S., Daultani, Y., Tiwari, M.K., Mahanty, B., 2015. Rule based optimization for a bulk handling port operations. Journal of Intelligent Manufacturing doi: 10.1007/s10845-015-1108-7.

Purnomo, H.D., Wee, H.M., Rau, H., 2013. Two-sided assembly lines balancing with assignment restrictions. Mathematical and Computer Modelling 57, 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, 1191-1210.

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

Scholl, A., Boysen, N., 2009. Designing parallel assembly lines with split workplaces: Model and optimization procedure. International Journal of Production Economics 119, 90-100.

Simaria, A.S., Vilarinho, P.M., 2009. 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. Computers & Industrial Engineering 56, 489-506.

Sörensen, K., 2015. Metaheuristics-the metaphor exposed. International Transactions in Operational Research 22, 3-18.

Tasan, S., Tunali, S., 2008. A review of the current applications of genetic algorithms in assembly line balancing. Journal of Intelligent Manufacturing 19, 49-69.

Thomopoulos, N.T., 1967. Line Balancing-Sequencing for Mixed-Model Assembly. Management Science 14, B-59-B-75.

Tiwari, M.K., Vidyarthi, N.K., 2000. Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach. International Journal of Production Research 38, 3357-3384.

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, 1009-1015.

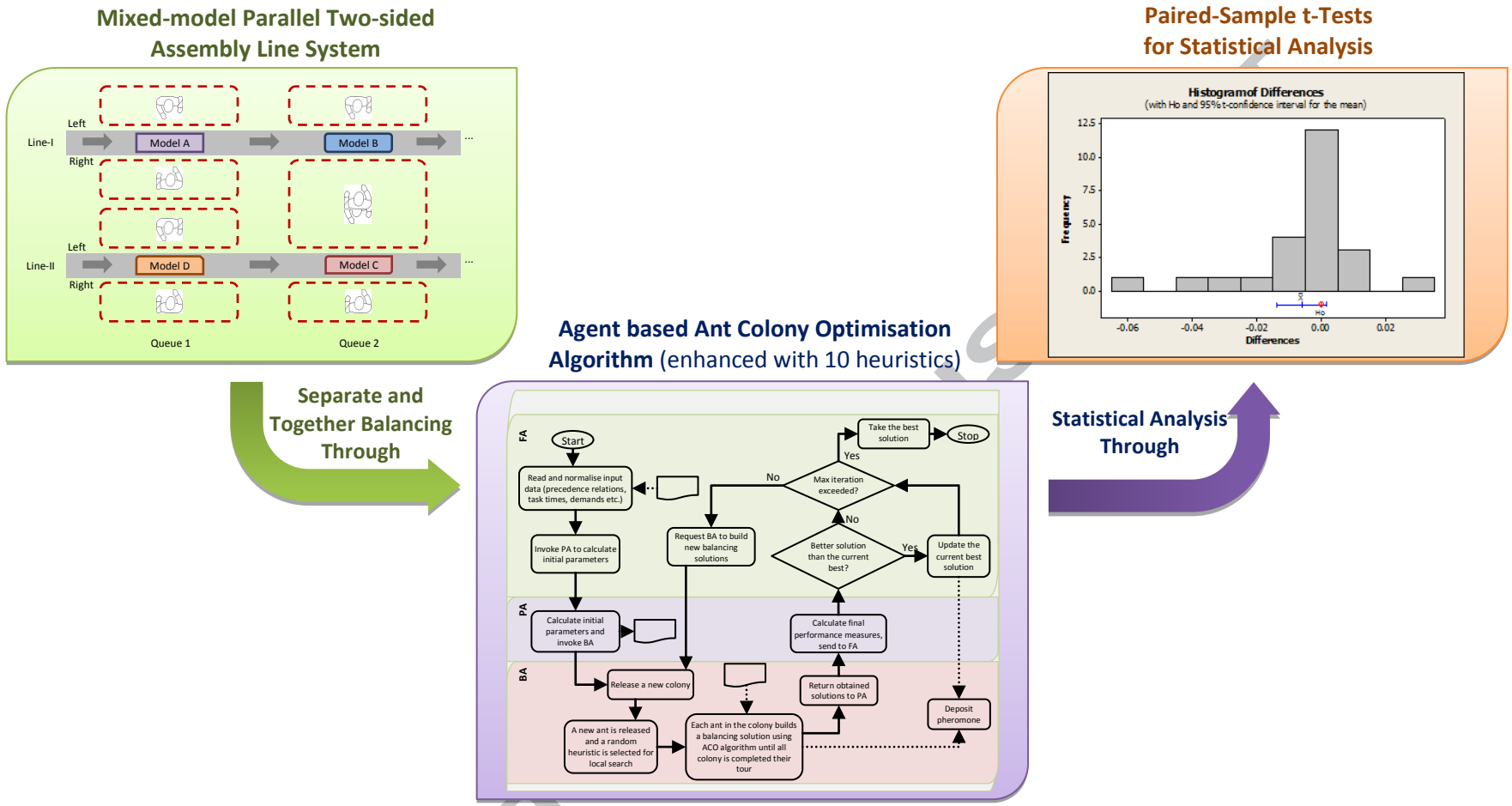
Xu, W., Xiao, T., 2011. Strategic Robust Mixed Model Assembly Line Balancing Based on Scenario Planning. Tsinghua Science and Technology 16, 308-314.

Yagmahan, B., 2011. Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. Expert Systems with Applications 38, 12453-12461.

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, 225-241.

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



Highlights

- Mixed-model parallel two-sided assembly line balancing problem (MPTALBP) is studied
- Agent based ant colony approach enhanced with 10 heuristics is developed
- Ants have opportunity to randomly select one of those heuristic search behaviours
- A new modified lower bound formulation is proposed for MPTALBP
- Statistical tests prove the benefits of the proposed system and the algorithm