Accepted Manuscript

Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines

Ibrahim Kucukkoc, David Z. Zhang

DOI : 10.1007/s00170-015-7320-y
Final Version : http://link.springer.com/article/10.1007/s00170-015-7320-y

To appear in : *The International Journal of Advanced Manufacturing Technology*
ISSN : ISSN 0268-3768

Please cite this article as:
Kucukkoc, I., Zhang, D.Z., Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines, *The International Journal of Advanced Manufacturing Technology* (2015), doi: http://dx.doi.org/10.1007/s00170-015-7320-y

The International Journal of Advanced Manufacturing Technology

Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines

Ibrahim Kucukkoc[1,2,*], David Z. Zhang[1]

[1] College of Engineering, Mathematics and Physical Sciences, University of Exeter, Streatham Campus, Exeter, England, UK
[2] Department of Industrial Engineering, Balikesir University, Cagis Campus, Balikesir, Turkey

Different from a large number of existing studies in the literature, this paper addresses two important issues in managing production lines, the problems of line balancing and model sequencing, concurrently. A novel hybrid agent based ant colony optimization – genetic algorithm approach is developed for the solution of mixed-model parallel two-sided assembly line balancing and sequencing problem. The existing agent based ant colony optimization algorithm is enhanced with the integration of a new genetic algorithm based model sequencing mechanism. The algorithm provides ants the opportunity of selecting a random behavior among ten heuristics commonly used in the line balancing domain. A numerical example is given to illustrate the solution building procedure of the algorithm and the evolution of the chromosomes. The performance of the developed algorithm is also assessed through test problems and analysis of their solutions through a statistical test, namely Paired-Sample t-Test. In accordance with the test results, it is statistically proven that the integrated genetic algorithm based model sequencing engine helps agent based ant colony optimization algorithm robustly find significantly better quality solutions.

Keywords: assembly line balancing; model sequencing; mixed-model parallel two-sided assembly lines; agent based ant colony optimization; genetic algorithm; artificial intelligence.

1. Introduction

Assembly line is a flow-line production system and is usually comprised of a set of sequentially linked workstations in which a set of tasks is performed by operators. Since the first assembly line was utilized by Henry Ford and his colleagues in 1913, the design and implementation of assembly lines have been critical issues for practitioners and academics [1].

* *Corresponding author. Email: i.kucukkoc@exeter.ac.uk; ikucukkoc@balikesir.edu.tr*

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Assembly line balancing problem is determining the optimal partitioning of tasks to the workstations by satisfying certain constraints (*i.e.* capacity constraint, precedence relationship constraint) [2,3]. Additional constraints may also arise depending on the line configuration, model variation, assumptions used *etc.*

With an increasingly consumer-centric global market, single model lines, where only one model is produced on the line, come up short in satisfying growing trend for highly customized product variability. In this context, mixed-model production lines have become popular as a result of companies' high-mix/low-volume manufacturing strategies. Thus, manufacturers can produce different models of a base product on a mixed-model assembly line (which was introduced by Thomopoulos [4]) instead of constructing and maintaining a new line for each model [5].

Mainly two types of problems arise for mixed-model assembly lines: *line balancing problem* and *model sequencing problem*. While the decision of which task will be performed in which workstation is made in the line balancing problem, the model sequencing problem determines the production sequence of different product models assembled on the same line. These two problems are tightly interrelated to each other and must be handled together to obtain a successfully implemented mixed-model assembly line. This is particularly important if more than one mixed model line, which is constructed in parallel to each other, is balanced together. Kucukkoc and Zhang [6] illustratively showed the dependency of line balancing problem on model sequencing problem (and vice versa). However, many researches addressed these two problems separately (for instance, see Erel and Gokcen [7], Matanachai and Yano [8], Vilarinho and Simaria [9], McMullen and Tarasewich [10], Yagmahan [11], Hamta *et al.* [12], Kucukkoc *et al.* [13] for the line balancing problem; and Yano and Rachamadugu [14], Kim *et al.* [15], Zheng *et al.* [16], Bautista and Cano [17], Zhu *et al.* [18], Manavizadeh *et al.* [19] and Xu and Li [20] for the model sequencing problem) with various objectives. The reader can refer to Kucukkoc and Zhang [6] and Zhang and Kucukkoc [34] for a summary of main contributions on Mixed-model Assembly Line Balancing Problems (MALBPs) and to Boysen *et al.* [21] for a comprehensive survey on sequencing mixed-model assembly lines.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Based on the operation side utilization of the lines, assembly lines could be classified into two groups: (i) one-sided assembly lines and (ii) two-sided assembly lines [22]. Two-sided assembly lines -introduced by Bartholdi [23]- are chiefly used in the production of large-sized products and are more practical for large-sized products (*e.g.* buses and trucks) than for small ones (*e.g.* electrical drills). Workers at each pair of opposite stations work in parallel on different tasks but on the same individual item. Two-sided lines differ from one-sided lines by means of necessity to perform some tasks on a specific side (Left-L or Right-R) of the line and some can be performed on any of the sides (Either-E) [24].

Various solution techniques were proposed ever since the *parallel assembly line balancing problem* [25] and *two-sided assembly line balancing problem* [23] were introduced. Large numbers of researchers have addressed mixed-model lines, two-sided lines and parallel lines (where two lines are located in parallel to each other [25]) and related problems separately in the literature. On the other hand, few researchers dealt with the combination of any of those problems (*i.e.* mixed-model two-sided lines, parallel mixed-model lines, *etc.*) although such lines have wide application areas and are commonly encountered in industry. *Parallel two-sided assembly line balancing problem* was introduced by Ozcan *et al.* [26] and a Genetic Algorithm (GA) based solution technique was proposed by Kucukkoc and Zhang [22,27]. Kucukkoc and Zhang [28] balanced parallel-two sided assembly lines using ant colony optimization algorithm with the aim of minimizing two conflicting objectives: cycle time and total number of workstations. Simaria and Vilarinho [29], Ozcan and Toklu [30] and Chutima and Chimklai [31] dealt with *mixed-model two-sided assembly line balancing problem* and developed different approaches; namely ant colony optimization, simulated annealing and particle swarm optimization, respectively. Nevertheless, *parallel mixed-model assembly lines* were studied by Ozcan *et al.* [32] only. The study of Ozcan *et al.* [32] differentiate from others as they consider the model sequencing problem as well as the line balancing problem. Please refer to Battaïa and Dolgui [33] for a comprehensive taxonomy of those problem types and their solution approaches.

Zhang and Kucukkoc [34] defined the *Mixed-model Parallel Two-sided Assembly Line Balancing Problem*, which is comprised of complex characteristics of different line configurations: mixed-model

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

lines, parallel lines, and two-sided lines. Kucukkoc and Zhang [35] developed an agent based ACO approach for the general solution of the problem, which is independent from the launched model sequence. Kucukkoc and Zhang [6] improved it by taking the model sequencing problem into consideration along with the line balancing problem and introduced the *Mixed-model Parallel Two-sided Assembly Line Balancing and Sequencing (MPTALB/S)* problem. Framework of a possible solution approach, Agent Based Ant Colony Optimization Algorithm (called ABACO/S), was also reported by Kucukkoc and Zhang [6] with no experimental results.

Another research was carried out by Kucukkoc and Zhang [36] (i) to mathematically model the MPTALB/S problem, (ii) to test the performance of the previously proposed ABACO/S algorithm, and (iii) to comparatively show the significance of solving line balancing and model sequencing problems together. In their research, an agent deploys ant colonies to build balancing solutions for appointed model sequences. The user is allowed to choose between two different schemes provided for the determination of model sequences: (i) combinatorial sequencing, and (ii) random sequencing. If combinatorial sequencing is chosen, the algorithm generates all possible model sequences for assembly lines and tries all model sequence combinations one by one. If the second option is chosen by the user, the algorithm tries a user defined number of random model sequences. While the former option increases the possibility of obtaining a well-balanced solution, the latter one returns solutions faster than the other. Therefore, a model-sequencing mechanism is needed to be able to obtain robust solutions with less effort. With this motivation, a GA based model sequencing procedure is integrated to the ABACO/S and a new hybrid Agent Based Ant Colony Optimization – Genetic Algorithm (ABACO/S-GA) approach is proposed for the solution of MPTALB/S problem in the current work.

Several real world characteristics of manufacturing systems could be of interest for assembly lines to have more realistic as well as sustainable designs [37]. However, due to the sophisticated nature of the studied problem in this research, some assumptions have been made to keep its complexity at a minimum level and make it solvable using today's computerized technology. The most important of these assumptions are as follows:

- Operators are multi-skilled and they have no preference(s) on tasks and/or models.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

- Task processing times may differ from one model to another but not from one workstation to another.
- Demand is known and deterministic in a planning horizon.

In the literature, the studies which attempt to relax such assumptions have considered relaxing only a limited number of them at a time and on relatively simple assembly line configurations. Also, such studies have not considered both of the line balancing and model sequencing problems, each of which is an NP-hard class of combinatorial optimization problem, at the same time. For example, Song *et al.* [38] proposed an approach to balance a straight production line through optimal allocation of operators considering operator efficiency. They comparatively showed that the proposed optimization method outperforms the industry practice. Corominas *et al.* [39] considered temporary and permanent operators, referred to as skilled and un-skilled workers, for rebalancing of a motorcycle assembly line. Manavizadeh *et al.* [40] considered operator skill levels on mixed-model U-shaped assembly line system allowing two types of operators: permanent and temporary. The balancing problem was solved and the number of stations required was determined first. Second, workers – who were classified into four types based on their skill levels - were assigned to the workstations in which they are qualified to work. A general framework to model skill requirements and skill conditions for assembly line balancing configurations was provided by Koltai and Tatay [41]. They defined three types of skill constraints, *i.e.* low skill, high skill and exclusive skill; and summarized their mathematical descriptions on simple assembly line balancing models. Also, Fattahi *et al.* [42] addressed multi-manned workstations where a group of workers can perform tasks simultaneously on the same item. When the resource dependent task times are applied, not only tasks but also resource alternatives (number of workers and equipment type) are assigned to the workstations [43]. Kara *et al.* [44] addressed resource-dependent task times on straight and U-shaped assembly lines and developed integer programming formulations. Jayaswal and Agarwal [43] considered resource dependent task times in balancing U-shaped assembly lines and proposed a simulated annealing approach for the solution of the problem. As could be seen, it is clear that none of these articles have considered both line balancing and model sequencing problems simultaneously. However, as it will be mentioned in

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Section 6, such assumptions as listed above are the limitations of this work and are left to future study which we believe that will be built on the current work. Furthermore, tasks may have different processing times based on their processing sequences in the workstations as in the study of Gajpal *et al.* [45] for flow-shop scheduling problem.

The remainder of this article is organized as follows. In Section 2, the characteristics of the MPTALB/S problem are presented with the notations and assumptions. Section 3 depicts the proposed ABACO/S-GA solution method by illustrating its component architecture and the main module characteristics. In Section 4, the solution building procedure of the algorithm is simulated through a numerical example. The experimental results of quantitative analysis of the proposed method are reported in Section 5 and, finally in Section 6, the conclusions on the research and outcomes achieved are drawn.

2. Balancing and sequencing of mixed-model parallel two-sided lines

The main characteristics of the problem addressed in this research (MPTALB/S problem) will be provided briefly in this section along with the assumptions made. The problem has already been defined by Kucukkoc and Zhang [6] and formulated mathematically by Kucukkoc and Zhang [36].

2.1. Notation

The following notations will be used for describing the problem characteristics:

L_h : The h^{th} line $(h = 1, \dots, H)$,

m_{hj} : The j^{th} product model on line L_h $(j = 1, \dots, M_h)$, where M_h is the number of product models made on line L_h,

x : Side of the line, $x = \begin{cases} 0 & \text{indicates left side of relevant line} \\ 1 & \text{indicates right side of relevant line} \end{cases}$,

W_{hkx} : The k^{th} workstation on line L_h $(k = 1, \dots, K_h;\ x = 0, 1)$, where K_h is total number of workstations on line L_h,

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

t_{hji} : The i^{th} task for model m_{hj} on line L_h $(i = 1, \dots, T_{hj})$, where T_{hj} is total number of tasks for model m_{hj} on line L_h,

P : A pre-specified planning period,

D_{hj} : Demand, over the planning period, for model m_{hj} produced on line L_h,

C_h : Cycle time of line L_h,

C : Common cycle time for all lines, in other words least common multiple of cycle times belonging to each line $(C = LCM(C_1, \dots, C_H))$,

MPS_h : Minimum part set or model mix of line L_h $(d_h = d_{h1}, \dots, d_{hM_h})$,

cd_h : Greatest common divisor of product model demands (D_{hj}) for line L_h,

d_{hj} : Normalized demand for model m_{hj} in model mix of line L_h, where a normalized demand for a product model is defined as the demand in terms of greatest common divisor of the relevant line,

MS_h : Model sequence of line L_h,

S_h : Total number of product models on line L_h for one MPS_h (the length of MS_h for one MPS_h), $\left(S_h = \sum_{j=1}^{M_h} d_{hj}\right)$,

$LCM(S_1, \dots, S_H)$: Least common multiple of S_h values $(h = 1, \dots, H)$,

φ : Production cycle $(\varphi = 1, \dots, \phi)$, where $\phi = LCM(S_1, \dots, S_H)$,

TS_h : The total number of possible model sequences for a mixed-model line (L_h).

2.2. Problem characteristics

A Mixed-model Parallel Two-sided Assembly Line (MPTAL) system consists of more than one two-sided assembly line (symbolized with L_h $(h = 1, \dots, H)$) located in parallel to each other and at least two similar models of a product (where models are symbolized with m_{hj} $(j = 1, \dots, M_h)$) are produced on each of the lines (see Figure 1). A workstation is represented with W_{hkx} $(k = 1, \dots, K_h;\ x = 0, 1;$

where x is a binary variable and '0' and '1' symbolize left side and right side of the line, respectively) and workstations are designated on both sides (Left and Right) of each of the lines.

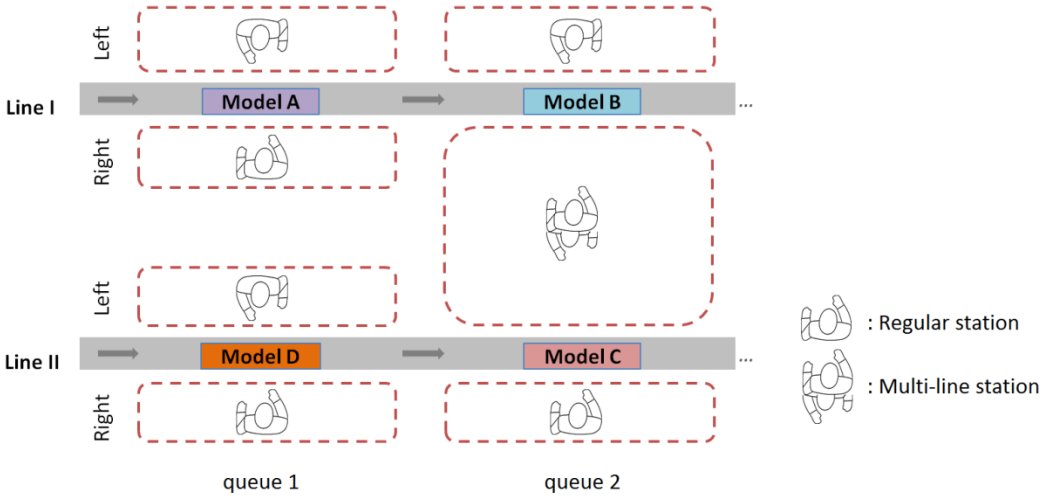

Figure 1. Mixed-model parallel two-sided assembly line [6].

Each model has its own set of tasks, t_{hji} $(i = 1, \dots, T_{hj})$, that need to be performed by operators located in workstations by taking precedence relationships into consideration. A common precedence relationship diagram is constituted for product models assembled on the same line and P_{hji} holds the set of predecessors of task t_{hji} for model m_{hj} on line L_h. By this way, common tasks between different models are assigned to the same workstation to minimize machinery costs and maximize the benefits of tool sharing. Processing time of a task is represented with pt_{hji} and each task must exactly be assigned to one workstation in a feasible solution. The sum of processing times of all tasks assigned to a station constitutes its workload and the workload of any workstation cannot exceed the designated cycle time for this workstation. In MPTALs, the cycle time (C_h) of each line may be different from each other. C_h is calculated according to demand over the planning horizon for each line as follows:

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

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

However, a common cycle time should be determined to avoid conflicts in multi-line stations that may arise due to cycle time differences. For this aim, Least Common Multiple (LCM) of cycle times

[25] is designated as common cycle time (C) and task times are normalized according to the ratio of original cycle time to common cycle time. This will be exemplified in Section 4 but the reader may refer to Gökçen *et al.* [25] for detailed explanation.

To tackle the model sequencing problem, the Minimum Part Set (MPS) principle of Bard *et al.* [46] is used [32]. According to this approach the MPS on line L_h (MPS_h) is calculated by dividing total demands of models by the greatest common divisor of these demands. Let the greatest common divisor of D_{hj} $(j = 1, \dots, M_h)$ be represented by cd_h $(h = 1, \dots, H)$ and model mix of line L_h be denoted by the vector $d_h = (d_{h1}, \dots, d_{hM_h})$, where $(h = 1, \dots, H)$.

$$d_{hj} = \frac{D_{hj}}{cd_h}; \quad j = 1, \dots, M_h;\ h = 1, \dots, H\ . \tag{2}$$

Regardless of the other model sequences, the model sequence of line L_h is represented by MS_h. The length of MS_h for one MPS_h also determines the total number of products on line L_h for one MPS_h and is represented by S_h ($S_h = \sum_{j=1}^{M_h} d_{hj}$). The length of the MS_h affects the number of different model combinations seen on the lines and determines how many different product cycles ($\varphi = 1, \dots, \phi$) the system should be split into. Total number of production cycles (maximum number of model show-ups), ($\phi = MS_{max}$), which may appear at a cycle can be calculated as follows:

$$MS_{max} = LCM(S_1, \dots, S_H); \qquad h = 1, \dots, H\ . \tag{3}$$

The total number of possible model sequences for a mixed-model line is computed using the equation given below [19]:

$$TS_h = \frac{\left(\sum_{j=1}^{M_h} d_{hj}\right)!}{\prod_{j=1}^{M_h}(d_{hj}!)} \tag{4}$$

But, when two mixed-model assembly lines are balanced together (as in here), the number of sequences emerging for the system could be computed by multiplying total number of sequences belonging to each of the lines ($TS_1 \times TS_2$). An exemplification of these calculations will be provided with an example in Section 4.

2.3. Constraints and assumptions

The line system considered in this paper operates under the following conditions [36]:

- Each task for each product model must be assigned to exactly one workstation.
- Due to some technological or organizational constraints, tasks can be assigned to only a predetermined operation side (L or R) or either (E) side.
- For a task to be assigned, all of its predecessors must have been assigned and completed.
- To maximize resource utilization, common tasks between models on the same line are forced to be assigned to the same workstation. Therefore, a joint precedence diagram is constituted for the product models produced on the same line and is used for task assignment.
- Processing times of tasks may differ between similar models but they are known and deterministic (when a task is not required for a product model, its processing time is assumed zero). There is no variation in task processing times depending on the resources that they are assigned.
- Total workload time of a workstation must be equal to or lower than the designated cycle time.
- Two or more similar models of a product are assembled on each of the two or more parallel two-sided assembly lines.
- In a planning horizon, demands for different models are known and deterministic.
- Only one operator can be assigned to a workstation.
- Operators are multi-skilled and do not have preferences on workstations and operation sides.
- Lines start new operations as soon as they finish previous operations.
- Work in process inventory is not allowed, and operator travel times are ignored.

3. The proposed method

ACO and GA are well-known and widely applied meta-heuristics in solving combinatorial optimization problems, and engineering optimization problems in particular. While ACO mimics the foraging behavior of real ant colonies in nature, GA mimics the natural selection and evolution of individuals [47,48]. The basic principles of ACO and GA were laid down by Dorigo *et al*. [49] and Holland [50], respectively. Since then, these two meta-heuristics have attracted researchers and been

enhanced in terms of solution capacity and efficiency. Agents are program scripts interact with each other to solve complex problems that are beyond their individual capabilities and have been utilized widely to solve complex manufacturing problems [51,52]. Recent advances could be found from Cordon *et al.* [53], McMullen and Tarasewic [10] and Blum [54] for the ACO technique; Srinivas and Patnaik [55] and Li *et al.* [56] for the GA technique; and Leitato *et al.* [57] for the agent based techniques on manufacturing control. Specifically, Tasan and Tunali [58] reviewed current applications of GAs in assembly line balancing domain.

ACO and GA have been hybridized successfully for a wide range problems, such as in Lee *et al.* [59], Chen and Chien [60], Li *et al.* [61], and Akpinar *et al.* [5]. Among these hybridization techniques, the method of Akpinar *et al.* [5] was used to solve MALBP with sequence dependent setup times between tasks. The algorithm proposed by Akpinar *et al.* [5] aims at enhancing the solution building procedure of ant colony optimization by incorporating GA as a local search strategy. Based on this motivation, GA is integrated to the ABACO/S to enhance the solution capacity of the algorithm while decreasing the computational effort. In the following sub-sections, we describe the ABACO/S-GA approach developed in this research.

3.1. Outline

ABACO/S-GA consists of four-level agents: Planning Agent (PA), Facilitator Agent (FA), Sequencing Agent (SA), and Balancing Agent (BA), which interact with each other to solve the problem collectively. The algorithm systematically searches for smaller number of workstations and shorter line length to solve the MPTALB/S problem for different candidate model sequences generated by the integrated GA mechanism in SA. For each candidate model sequence, ACO algorithm tries to find the best task assignment. The outline of the developed algorithm is depicted in Figure 2.

The input data (e.g. task times, precedence relationships, model demands, *etc.*), algorithmic parameters (*i.e.* ACO parameters such as α, β, ρ, *etc.*; GA parameters such as population size, crossover rate, mutation rate, *etc.*), and user preferences (such as objective function weights) are read

by the FA and the algorithm is initialized. Then, intermediate parameters (*i.e.* greatest common divisors of model demands, least common multiple of cycle times) are calculated and data is processed (*i.e.* task times are normalized, common cycle time is calculated, minimum part sets are determined, *etc.*) to make it ready for use by other program components. Initial population is generated by SA, where each chromosome represents a complete model sequence. SA requests fitness evaluation of the chromosomes in the population from the BA. BA employs ACO to build balancing solutions for given chromosomes and returns fitness values to SA. Genetic operators perform crossover and mutation for the selected chromosomes and the fitness values of the newly obtained children and mutants after these processes are evaluated by the BA again.

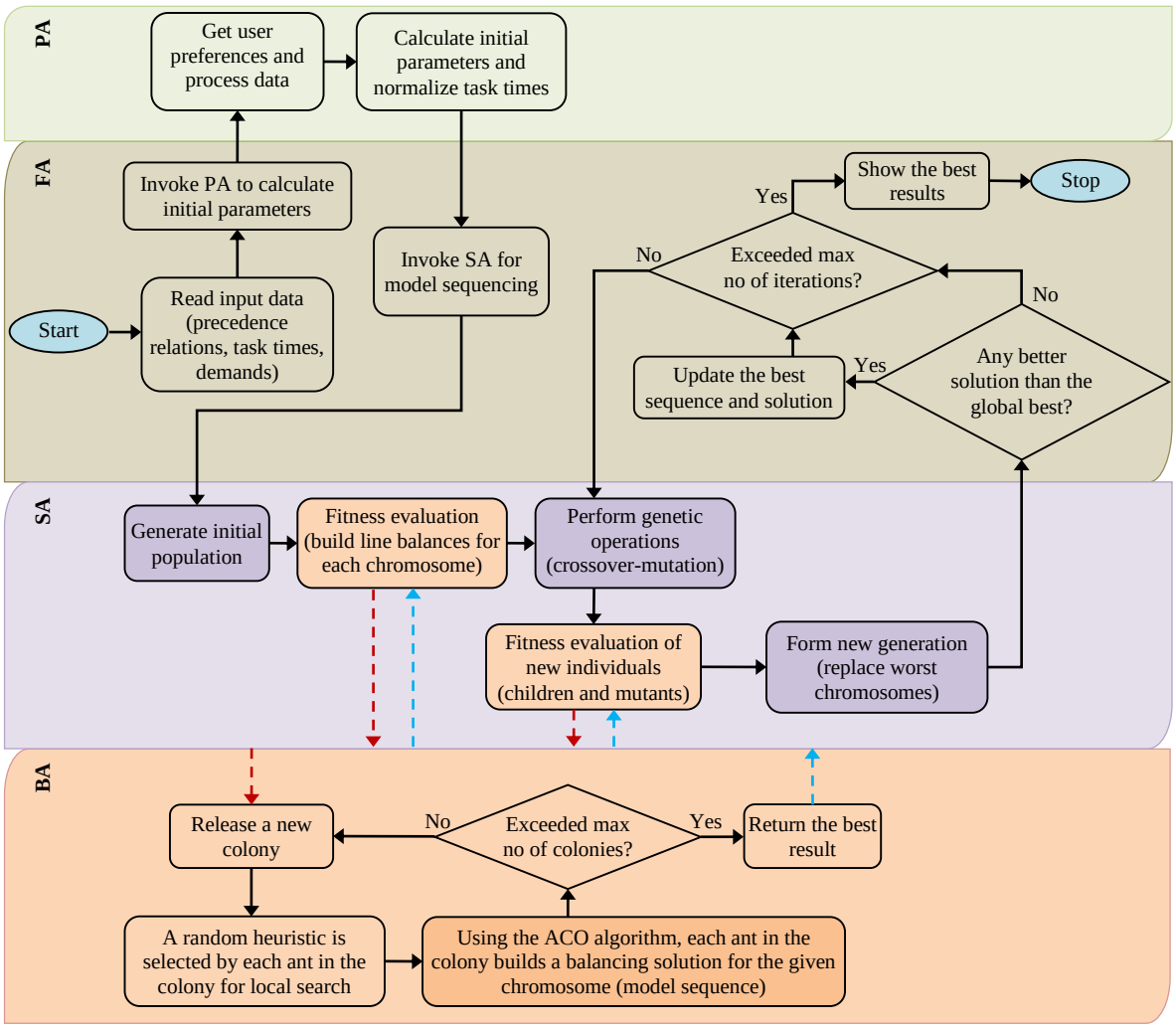

Figure 2. The outline of the proposed ABACO/S-GA approach.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

The new generation is formed by replacing the worst chromosomes with the best ones among the children and mutants (if any). Genetic operators are performed again for the selected individuals in the population and the new generation is formed in accordance with their fitness values. This cycle is repeated until a pre-determined number of iterations are exceeded. The best solution is updated if a better solution is found during this cycle and is shown when the algorithm stops. The procedures of the GA (for model sequencing) and ACO (for line balancing) are explained in the following sub-sections in detail.

3.2. GA for model sequencing

The simulation of the employed GA is given in Figure 3. Initial population is generated by building up feasible chromosomes randomly, considering the number of chromosomes that the population must have (population size).

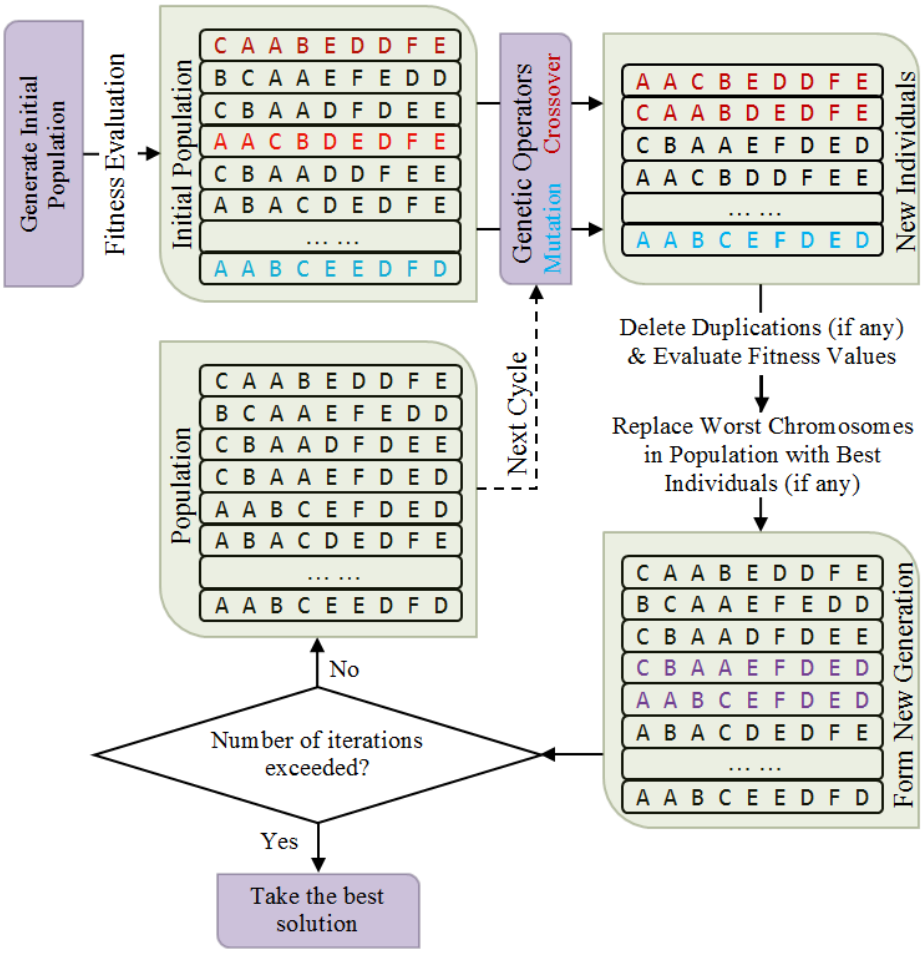

Figure 3. Simulation of the employed genetic algorithm.

A chromosome is made up with the ordered model types according to the demand and minimum part sets calculated. To build a feasible chromosome, model types that belong to Line I are located to the head of the chromosome, while model types for Line II are allocated to the tail of the chromosome. Thus, chromosome length equals to the sum of model sequences length produced on the lines and each model appears on the chromosome as the number of times it shows up in the model sequence (MS_h). A chromosome sample for a given model sequence of $MS_1 = AABC$ and $MS_2 = DDEEF$ is exhibited in Figure 4.

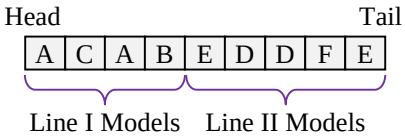

Figure 4. Representation of a real-coded chromosome

For the evolution of individuals, one-point crossover and two-gene mutation operators are applied on randomly selected chromosomes from the population. Figure 5 represents the working procedure of crossover procedure used in the algorithm. The place where the models belonging to Line I finishes and the models of Line II starts is determined as the cutting point of the chromosomes and the head and tail parts of the parent chromosomes are matched crosswise to acquire new off-springs.

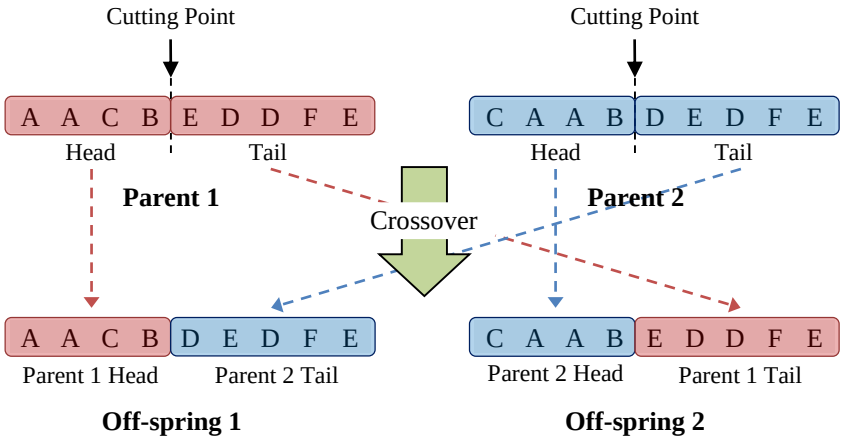

Figure 5. Single point crossover procedure

Mutation procedure is conducted by swapping two randomly selected genes within the same zone. In other words, if a gene is selected before (after) the border, it is swapped by a randomly selected gene before (after) the border (see Figure 6).

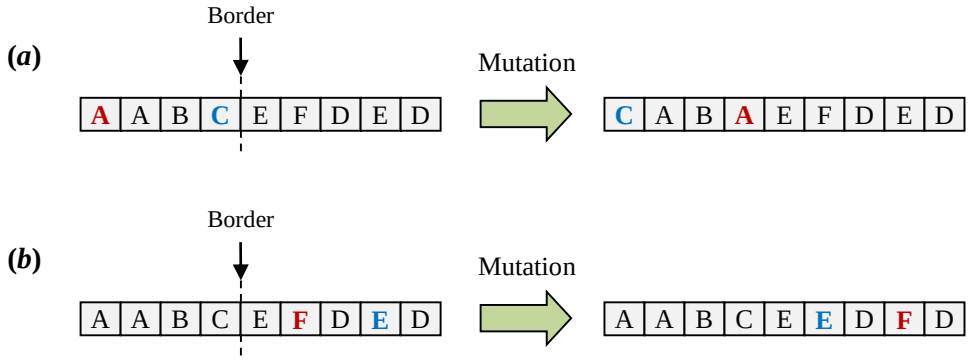

Figure 6. Mutation with random two genes exchange within (a): first zone, and (b): second zone.

3.3. ACO for line balancing

Fitness values of chromosomes in the population and of new individuals obtained from the genetic operators are computed by the BA using ACO algorithm. This algorithm builds line balancing solutions for the model sequences using the procedures outlined in Figure 7. Each ant in the colony builds a balancing solution (using the procedure given in Figure 8, where $st(k)$ and $st(\underline{k})$ represent station time of the current station and its mated station, respectively) and the performance measures are evaluated according to the quality of the obtained solution.

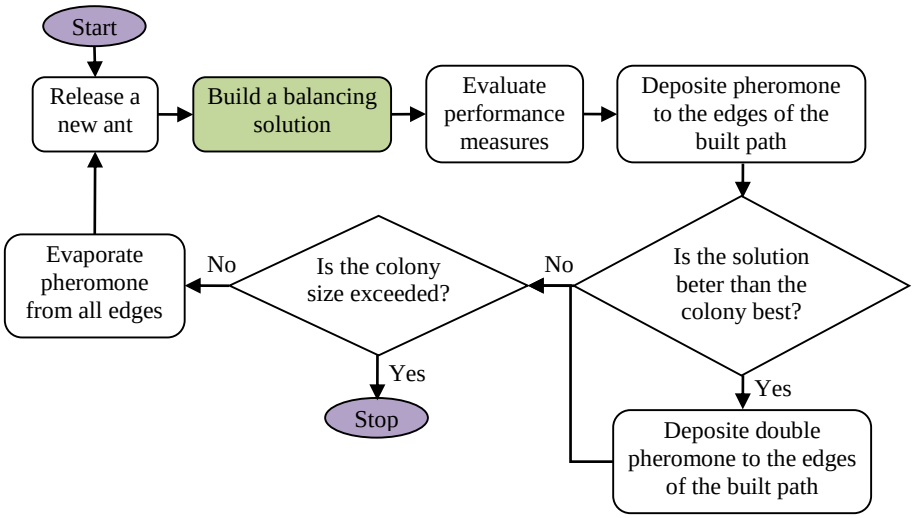

Figure 7. Ant colony optimization procedure

As could be seen from Figure 8, the idea lying behind the balancing solution procedure is to determine available tasks for the current situation and to select and assign tasks one-by-one to the current workstation. Solution building procedure starts from a randomly selected line and side and forwards

by selecting and assigning tasks (by ants) from the available tasks lists to the current position (please see Figure 9 for the procedure of determining available tasks for current position).

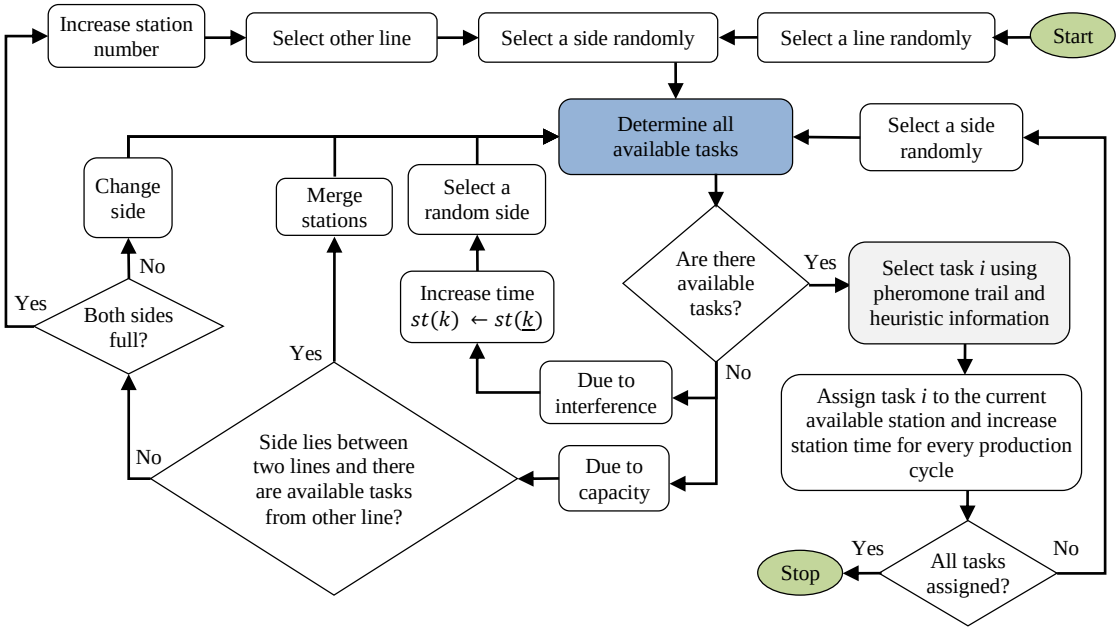

Figure 8. Building a balancing solution procedure.

Sometimes there may not be available tasks even all tasks are not assigned yet (see Figure 8). If this is due to the incomplete tasks on the other side of the two-sided line (this situation is called interference), $st(k)$ is increased to $st(\underline{k})$ and assignment process continues on a randomly selected side. If the reason is that there is not enough remaining capacity to perform any task from the current line but from the adjacent line, then the workstations are merged and a multi-line station is constructed to perform tasks from the opposite side of the other line.

When the balancing solution is obtained, the pheromone is laid between tasks and workstations on the edges of the drawn path according to the performance measures. Double amount of pheromone is released when a solution which is better than the best solution in the colony is found. By this way, best solutions are made favorable to be selected by following ants. The pheromone update rule is as follows:

$$\tau_{ik} \leftarrow (1-\rho)\tau_{ik} + \Delta\tau_{ik}\,, \tag{5}$$

where ρ and τ_{ik} represent the evaporation rate and the amount of virtual pheromone between task – workstation, respectively; $\Delta\tau_{ik} = Q/Performance\ measure$, and Q is a user determined parameter that effects the amount of pheromone deposited.

The selection probability of a task is computed as follows [36]:

$$p_{ik} = \frac{[\tau_{ik}]^{\alpha}[\eta_i]^{\beta}}{\sum_{y\epsilon Z_i}[\tau_{iy}]^{\alpha}[\eta_i]^{\beta}}, \quad (6)$$

where i, k, and Z_i indicate task, current workstation, and list of candidate tasks when task i is assigned, respectively. τ_{ik} and η_i are the amount of virtual pheromone between task – workstation, and the heuristic information of task i that comes from the randomly selected heuristic by each ant. To provide heuristic information and increase the local search capacity of the algorithm, ten heuristics are available to be selected by each ant: Computer Method of Sequencing Operations for Assembly Lines - Comsoal [62], Ranked Positional Weight Method – RPWM [63], Reverse Ranked Positional Weight Method – RRPWM (produced from RPWM), Longest Processing Time – LPT [64], Shortest Processing Time – SPT [65], Smallest Task Number – STN [66], Maximum Number of Predecessors – MNP (produced from Baykasoglu [65]), Least Number of Predecessors – LNP, (produced from MNP), Maximum Number of Successors – MNS (produced from Tonge [67]), Least Number of Successors – LNS (produced from MNS).

Ants can change their operation sides at any time to increase the possibility of obtaining diversified solutions. However, changing line is only possible when the capacities of current mated stations are full or there is no available task to assign to the current mated stations. Multi-line stations can be utilized between two adjacent lines by assigning tasks from the contrary side of the adjacent line if the capacity of the workstation is not full and there is at least one available task to be assigned.

The procedure of determining available tasks (given in Figure 9) plays a significant role in the overall balancing and sequencing system, because the solution that will be obtained at the end of this procedure must be feasible in terms of different model sequences, which change at every production cycle. Processing time of a candidate task is considered according to the actual model at the launched cycle, and processing time of a task for the relevant model must be equal to or less than the remaining

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

capacity to be able to assign it. Furthermore, workloads of workstations and earliest starting times of tasks must be watched carefully for each production cycle. This data is used to calculate remaining capacity of a workstation when determining whether a task is available or not.

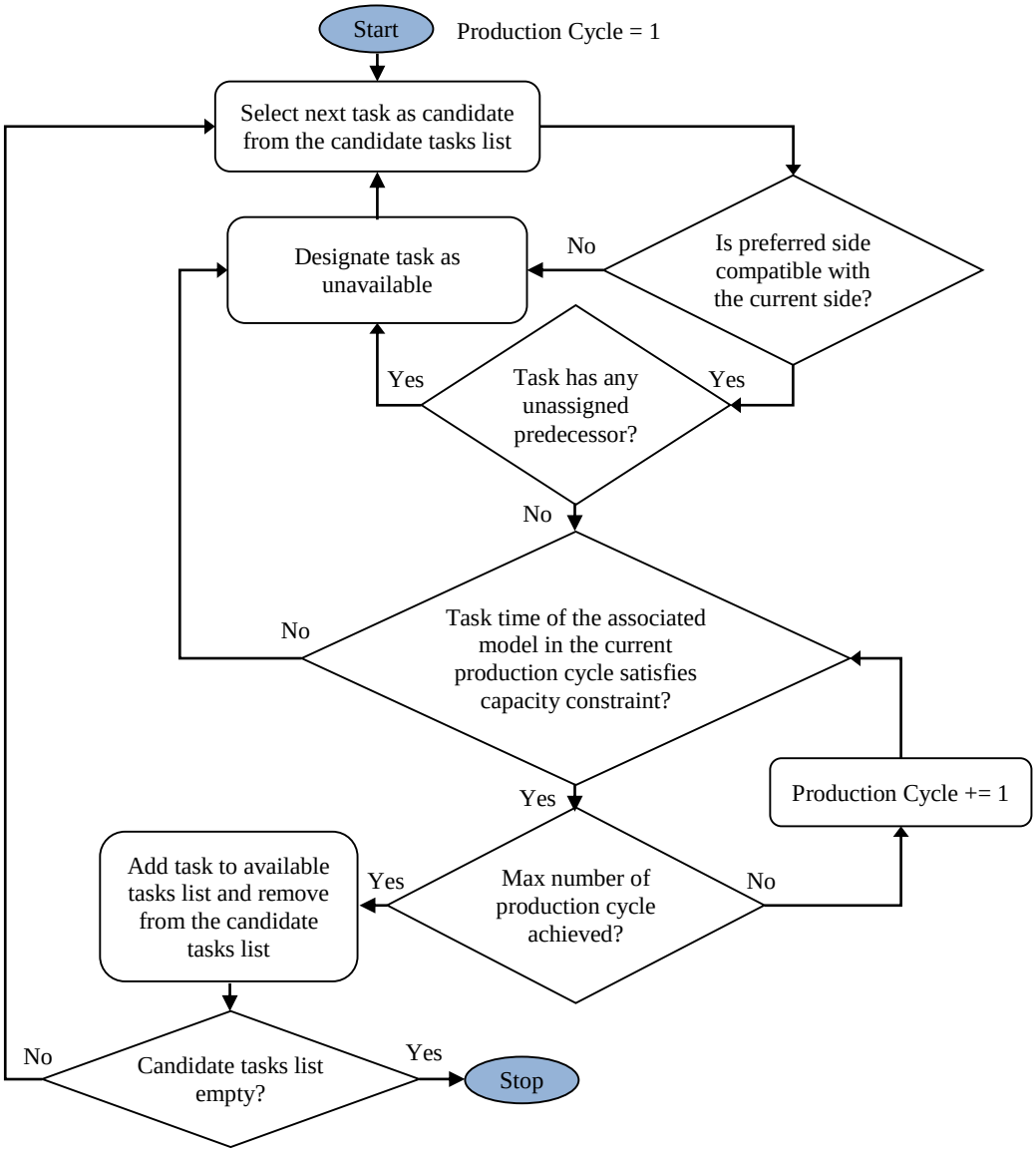

Figure 9. The procedure of determining available tasks.

3.4. Performance measure

The total space needed to perform production may become more important than the required *number of stations (NS)* if the manufacturer has certain space constraints. For that reason, *line length (LL)* is also considered as an additional objective to the number of stations in this research. Thus, the decision

maker could decide on the significance of the performance measures by simply changing the weighing parameters, γ_1 and γ_2, in the objective function:

$$Min\ Z = \gamma_1 LL + \gamma_2 NS\,. \tag{7}$$

4. Numerical example

An example is given in this section to illustrate the model sequencing procedure of the developed approach in detail.

4.1. Problem data

Problem data is taken from Kucukkoc and Zhang [36] and includes two different precedence relationship diagrams, P12 [68] and P16 [69], for two parallel two-sided assembly lines (P12 for Line I and P16 for Line II). Each of these diagrams is assumed common among three different models of a base product on each line: *i.e.* models A, B, and C on Line I; and models D, E, and F on Line II. Table 1(a) gives processing times of tasks with preferred operation directions (where 'L' means Left side, 'R' means Right side, and 'E' means Either side of the line) and their immediate predecessor tasks. When a task is not required to be performed for a specific model, its processing time is shown as '0' in the table. For a fixed planning horizon of 480 time units, demands are assumed $D_{1A} = 8$, $D_{1B} = 8$, $D_{1C} = 16$, $D_{2D} = 8$, $D_{2E} = 8$ and $D_{2F} = 8$.

4.2. Pre-processing

Cycle times could be computed as $C_1 = 15$ and $C_2 = 20$ time units, for Line I and Line II, respectively; and the LCM based approach, proposed by Gökçen *et al.* [25] and Ozcan *et al.* [32], is used due to the cycle time differences of the lines. $LCM(C_1, C_2) = 60$ is accepted as the common cycle time $(C = 60)$ for both lines and line divisors (ld_h) of the lines are obtained as $ld_1 = LCM(C_1, C_2)/C_1 = 60/15 = 4$ and $ld_2 = LCM(C_1, C_2)/C_2 = 60/20 = 3$. Task times of models on Line I and Line II are normalized by being multiplied with ld_1 and ld_2, respectively. Table 1(b) provides the problem data with normalized task times that will be used while balancing the lines.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Based on the model demands, minimum part sets are $MPS_1 = (1, 1, 2)$ and $MPS_2 = (1, 1, 1)$ for Line I and Line II, respectively. According to the minimum part sets on the lines, the number of possible model sequences for Line I and Line II are $TS_1 = 4!/(1! \times 1! \times 2!) = 12$ and $TS_2 = 3!/(1! \times 1! \times 1!) = 6$. This means $12 \times 6 = 72$ different combinations of model sequences must be tried if combinatorial sequencing is decided by the user. Moreover, the number of different production cycles subject to consideration for each model sequence combination is $CM(S_1, S_2) = LCM(4, 3) = 12$.

Table 1. Problem data for the given example: (a) before normalization and (b) after normalization.

(a)	**Line I (P12)**					**Line II (P16)**				
Task	Processing Time			Side	Immediate Predecessor(s)	Processing Time			Side	Immediate Predecessor(s)
	A	B	C			D	E	F		
1	4	2	6	L	-	5	7	4	E	-
2	8	10	7	R	-	0	4	0	E	-
3	3	5	3	E	-	5	10	7	L	1
4	0	2	4	L	1	4	8	2	E	1
5	3	1	2	E	2	3	4	8	R	2
6	1	6	0	L	3	1	2	3	L	3
7	2	0	2	E	4, 5	7	1	6	E	4, 5
8	5	6	6	R	5	4	4	5	E	6, 7
9	4	4	2	E	5, 6	2	2	1	R	7
10	2	5	0	E	7, 8	3	3	4	R	7
11	2	9	5	E	9	5	7	4	E	8
12	3	2	1	R	11	1	6	5	L	9
13	-	-	-	-	-	4	4	6	E	9, 10
14	-	-	-	-	-	5	2	3	E	11
15	-	-	-	-	-	0	4	1	E	11, 12
16	-	-	-	-	-	5	3	5	E	13

(b)	**Line I (P12)**					**Line II (P16)**				
Task	Processing Time			Side	Immediate Predecessor(s)	Processing Time			Side	Immediate Predecessor(s)
	A	B	C			D	E	F		
1	16	8	24	L	-	15	21	12	E	-
2	32	40	28	R	-	0	12	0	E	-
3	12	20	12	E	-	15	30	21	L	1
4	0	8	16	L	1	12	24	6	E	1
5	12	4	8	E	2	9	12	24	R	2
6	4	24	0	L	3	3	6	9	L	3
7	8	0	8	E	4, 5	21	3	18	E	4, 5
8	20	24	24	R	5	12	12	15	E	6, 7
9	16	16	8	E	5, 6	6	6	3	R	7
10	8	20	0	E	7, 8	9	9	12	R	7
11	8	36	20	E	9	15	21	12	E	8
12	12	8	4	R	11	3	18	15	L	9
13	-	-	-	-	-	12	12	18	E	9, 10
14	-	-	-	-	-	15	6	9	E	11
15	-	-	-	-	-	0	12	3	E	11, 12
16	-	-	-	-	-	15	9	15	E	13

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

4.3. Simulation of the solution procedure

The algorithm was run using the parameters $\alpha = 0.1$, $\beta = 0.2$, $\rho = 0.1$, $Q = 50$, $Initial\ Pheromone = 10$, $Colony\ Size = 10$, and $Number\ of\ Colonies = 5$ for the ACO; and $Population\ Size = 10$, $Crossover\ Rate = 0.4$, $Mutation\ Rate = 0.1$, and $Number\ of\ Iterations = 10$ for the GA. Initial population and evolution of the chromosomes for the first and the last generations of the population are briefly simulated in Table 2. The importance of line length in computing the fitness value is considered as double as of total number of utilized workstations ($\gamma_1 = 2, \gamma_2 = 1$).

The line balancing solutions are given for chromosomes in the initial population only. However, the fitness values of the chromosomes and the ones emerging from the crossover and mutation procedures are given for the first and the last iterations. The number of new chromosomes (offspring) obtained after crossover and mutation operations may differ from generation to generation as duplications are not allowed, and deleted if any. The numbers given in brackets in Table 2 symbolize the task assignments for the chromosomes given. Brackets separate tasks into groups for designated workstations on the lines. To explain it more, the task assignment configuration of the obtained best solution is depicted in Figure 10. Assignment order of tasks is symbolized with arrows.

Each ant starts assigning tasks from a randomly selected line and side. For this example, ant starts by assigning tasks belonging to models produced on Line I (A, B, and C) from side L, and selects task 3 to assign from the available tasks list of this position. Then, tasks 1, 4, and 6 are assigned one by one to this side before ant changes line side. Tasks 2, 5, and 7 are assigned to right side of the Line I and ant moves forward to Left side of Line II to assign tasks belonging to the models produced on this line (D, E, and F). Available tasks list is updated every time when a new task is assigned and if the capacity is full or there is not any available task to assign for the current line, line is changed and tasks are assigned to the new workstations on the new line. This process continues until all tasks are assigned.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Table 2. Initial population and evolution of the chromosomes through generations.

	Chromosome	Obtained Best Line Balance for the Chromosome	Fitness
INITIAL POPULATION	C A B C E F D	[[1, 6, 4], [3, 2], [2, 1], [5, 4, 7], [5, 9, 10], [8, 7, 12], [3, 6, 8], [9, 10, 13, 16], [], [11, 12], [11, 14, 15], []]	16
	C A B C F E D	[[1, 6, 4], [3, 2], [1, 4, 7], [2, 5, 9], [5, 11], [9, 7, 8], [3, 6, 12], [10, 13, 16, 8], [], [10, 12], [11, 15, 14], []]	16
	C C A B D F E	[[3, 1, 6, 4, 7], [2, 5], [2, 3, 6], [1, 4, 5, 7], [], [9, 11, 12], [8, 11, 13], [10, 14, 9], [], [8, 10], [16, 12, 15], []]	15
	C C A B F D E	[[1, 4, 3, 6], [2, 5, 7], [1, 4, 7], [2, 5, 10], [9, 11], [8, 10, 12], [3, 6, 8, 16], [9, 13], [], [], [11, 14, 12, 15], []]	15
	C A C B F D E	[[6, 1, 4], [3, 2], [2, 3, 6], [1, 5, 4, 7], [9, 11], [5, 8, 7, 10, 15], [8, 12, 16], [10, 9, 11, 13, 14], [], [12], [], []]	15
	C B C A D E F	[[1, 4, 3, 6, 7], [2, 5], [1, 3, 6], [2, 5, 4, 7, 10], [11], [9, 8, 10], [8, 12, 14, 12], [9, 11, 15, 13], [], [], [16], []]	15
	C A B C F D E	[[3, 1, 6, 4, 7], [2, 5], [4], [2, 1, 5, 7], [9, 11], [8, 10, 12], [3, 12, 6], [9, 10, 13, 16], [], [], [8, 11, 15, 14], []]	15
	C B A C D E F	[[3, 6, 1, 4, 7], [2, 5, 9], [1, 3, 6, 7], [2, 4, 5], [11], [8, 10, 12], [8, 11, 14, 12], [10, 9, 13, 16], [], [], [15], []]	15
	C A B C E F D	[[3, 1, 6, 4], [2, 5, 7], [1, 3, 6], [2, 4, 5, 7], [9, 10], [8, 11], [8, 12, 15, 16], [9, 10, 11, 13, 14], [], [12], [], []]	15
	C C A B E D F	[[3, 6, 1, 4, 7], [2, 5, 9], [4], [1, 2, 5, 7, 9], [11, 10], [8, 12], [12, 3, 6], [10, 13, 16], [], [], [8, 15], [11, 14]]	16

	…	…	…	…
	New Chromosomes after Crossover and Mutation	Fitness	Generation 1 After Replacement of Worst Chromosomes in the Population	Fitness
	C B A C F E D	15	C C A B D E F	12*
	C A B C D E F	15	C B A C F E D	15
	C C A B D E F	12	C C A B D F E	15
	C B C A F D E	16	C C A B F D E	15
	C A C B E F D	15	C A B C F D E	15
	C A B C F D E	15	C B C A D E F	15
	C C A B F D E	16	C A B C F D E	15
	C A B C D F E	15	C B A C D E F	15
			C A B C E F D	15
			C A B C D E F	15
	…	…	…	…
	New Chromosomes after Crossover and Mutation	Fitness	Generation 10 After the Replacement of Worst Chromosomes in the Population	Fitness
	C C A B F E D	16	C C A B D E F	12*
	C B A C D E F	15	C B A C F E D	15
	C A B C F E D	16	C C A B D E F	15
	C B A C F E D	15	C C A B D E F	15
	C A B C E F D	15	C A B C F D E	15
	A C B C F E D	15	A B C C D E F	15
	C C A B E D F	16	C A B C F E D	15
			C B A C F E D	15
			A C B C E F D	15
			C A B C D E F	15

**Best Chromosome: C C A B D E F (Model Sequences - Line I: CCAB, Line II: DEF)*

Best Line Balancing Solution:

[[3, 1, 4, 6], [2, 5, 7], [1, 3, 6], [2, 5, 4, 7, 9], [9, 11], [8, 10, 12], [12, 11, 15, 14], [8, 10, 13, 16]]

Best Fitness: 12 (LL=2, NS=8)

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Convergence of the algorithm for this example is exhibited in Figure 11 by means of objective value, line length and total number of required workstations. The algorithm was run for ten iterations and the change in the performance measures while generating the initial population was also recorded and represented between iterations #0-#1 in the figure.

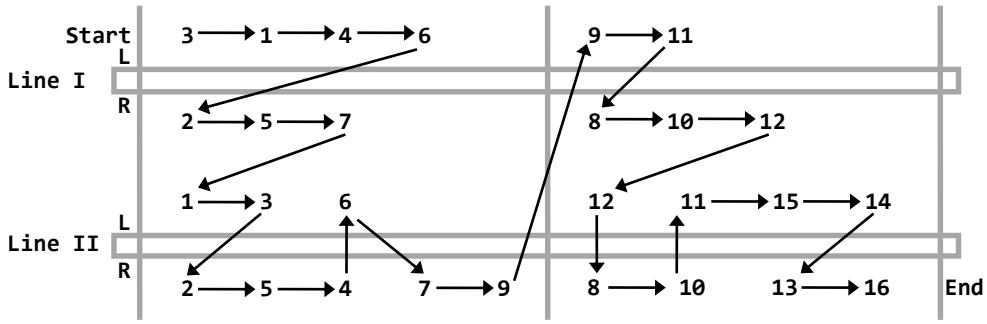

Figure 10. The task assignment configuration of the obtained best solution.

The algorithm calculates fitness values of the chromosomes in the initial population while generating them in iteration #0 and the best fitness value decreases from 16 to 15 at this stage. Then it converges quickly and finds the best solution after crossover and mutation operations in the first iteration. One of the offspring (CCABDEF) is designated as the best individual at this stage and this situation remains the same until the last iteration. Then the algorithm reaches to the maximum number of iterations at iteration #10 and is terminated with the best fitness value of 12. These graphs exhibit the effect of the model sequencing on the quality of the obtained line balance, once again.

5. Computational experiments

To assess the performance of the proposed approach, computational tests are performed and the results are compared with those existing in the literature. A Paired-samples t-Test is also conducted to statistically analyze the results obtained by the proposed approach.

The ABACO/S-GA algorithm is coded in Java™ SE 7u4 environment and run on a PC with a 3.1 GHz Intel Core™ i5-2400 CPU and 4GB of RAM. The parameters of the algorithm are chosen experimentally in accordance with the scale of the test case for a high quality solution and are given in Table 3. The values of these parameters may differ from one test problem to another to search the exponentially growing search space (with the increasing number of tasks and complexity) effectively.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

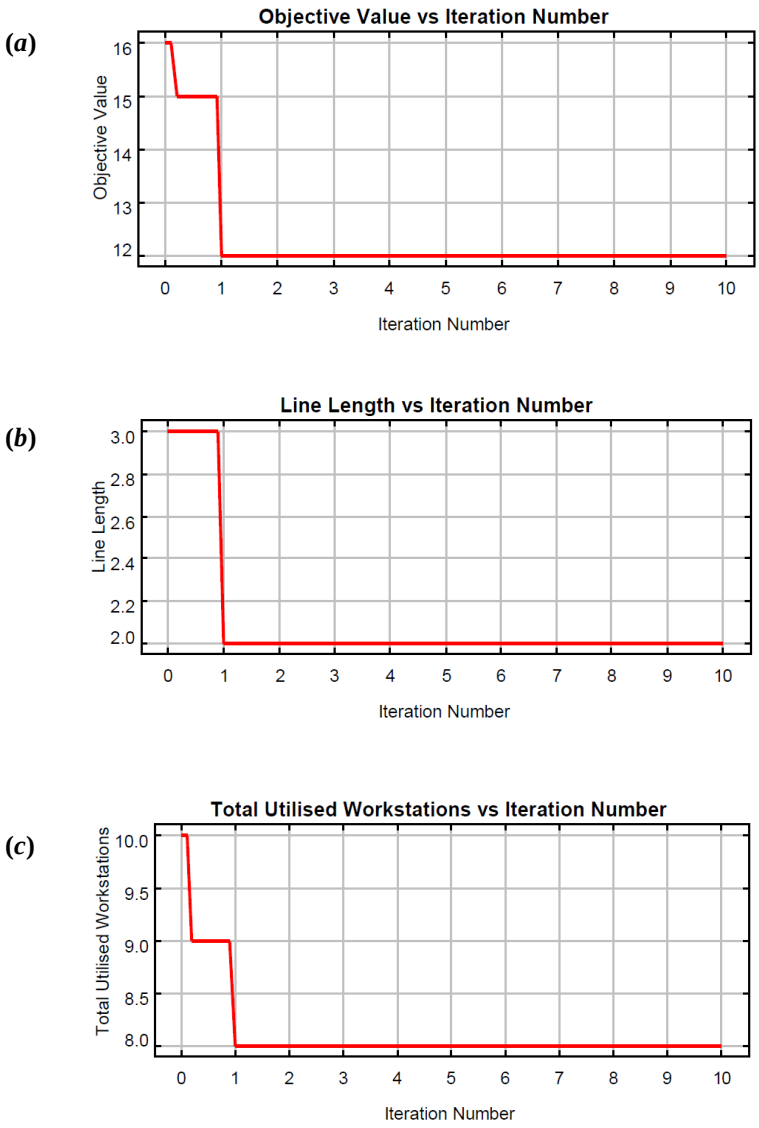

Figure 11. Convergence of the algorithm for the example problem by means of (a) objective value (fitness), (b) line length, and (c) total number of utilized workstations.

Table 3. Parameters of the hybrid ABACO/S-GA approach.

	Test Cases Stage 1	α	β	ρ	Q	Initial Pheromone	Colony Size	Number of Colonies
ACO	**1-9**	0.1	0.2	0.1	50	10	10	5
	10-30	0.1	0.2	0.1	50	15	20	10
	31-45	0.1	0.2	0.1	50	20	30	15

	Test Cases Stage 1	Population Size	Crossover Rate	Mutation Rate
GA	**1-9**	8	0.2	0.1
	10-30	10	0.4	0.2
	31-45	16	0.6	0.3

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Computational tests are carried out in two stages, namely Stage 1 and Stage 2, and the results are presented in the corresponding subsections. In Stage 1, the aim is to measure the efficiency of the ABACO/S-GA algorithm in terms of its capability to find the solution with less effort. Therefore, the algorithm is terminated when the target objective function value is found. The target objective value of a test case is set as the value obtained by ABACO/S [36] for that particular test case. In Stage 2, the aim is to measure the quality of the solutions gathered by ABACO/S-GA. In this stage, the algorithm is not terminated when a specific target value is achieved, instead it is run for a predetermined number of iterations and the solutions obtained by ABACO/S-GA are compared with those obtained by ABACO/S [36].

Test cases solved in this research (for which data are presented in Table 4 along with solution results of Stage 1) contain existing test cases in the literature as well as newly generated test cases for the current work (to remind, each test case is a combination of two test problems – one on each of the lines). So, a total of 20 test cases out of 45 test cases solved in this study are gathered from Kucukkoc and Zhang [36]. Kucukkoc and Zhang [35] generated test cases for mixed-model parallel two-sided assembly line balancing problem by combining well-known test problems in the literature. Then, Kucukkoc and Zhang [36] modified two of these test cases (see #10 and #19 in Table 4). Modified versions of these cases are considered and solved using the developed hybrid ABACO/S-GA algorithm to evaluate the solution building capacity of the proposed approach. To measure the performance of the proposed algorithm more comprehensively, new test problems (namely K20 and K36, see Appendices) are also generated and new test cases (see test cases marked with ‘N’ in Table 4) are built using the existing and newly generated test problems. As there are no results available in the literature for these particular test cases, the new test cases generated within the scope of the current study are also solved using the ABACO/S algorithm in the same conditions (e.g. using the same parameters) as in Kucukkoc and Zhang [36]. So that, a more comprehensive comparison can be made between ABACO/S and ABACO/S-GA procedures.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Table 4. Data for test cases and computational results – Stage 1.

Problem Data											**Computational Results**									
#Test Case	**Problem**		**Cycle Time**		**Demands L1**			**Demands L2**			**ABACO/S**					**ABACO/S-GA**				
	Line I	*Line II*	*Line I*	*Line II*	*A*	*B*	*C*	*D*	*E*	*F*	*LL*	*NS*	*OBJ*	*Best Sequence Line I-Line II*	*NI*	*LL*	*NS*	*OBJ*	*Best Sequence Line I-Line II*	*NI*
1	P9	P9	4	7	40	20	10	20	10	10	3	10	16	AABAABC-FDED	15	3	10	16	BAAACBA-DEFD	IP + 0
2	P9	P9	6	5	20	20	10	15	30	15	3	9	15	CAABB-EDFE	15	3	9	15	AABCB-EDEF	IP + 0
3[N]	P9	P9	7	7	10	15	15	20	10	10	2	7	11	ACCBACBB-DDEF	15	2	7	11	ACCBACBB-DDEF	IP + 0
4	P9	P12	5	8	40	20	20	20	20	10	3	8	14	AABC-DEFDE	15	3	8	14	BAAC-EDEDF	IP + 0
5	P9	P12	7	6	15	15	30	20	10	40	2	8	12	BCAC-DFFFFED	15	2	8	12	BACC-FEDFDFF	IP + 1
6[N]	P9	P12	6	7	15	10	10	10	10	10	2	8	12	AABCABC-EFD	15	2	8	12	CBAACBA-DFE	IP + 0
7	P12	P12	4	5	20	10	20	10	20	10	4	12	20	CCAAB-EDEF	15	4	12	20	BCCAA-EFED	IP + 1
8	P12	P12	6	5	20	10	20	30	15	15	3	10	16	ABACC-DFDE	15	3	10	16	CBCAA-EDFD	IP + 1
9[N]	P12	P12	7	8	20	40	20	20	30	20	2	7	11	CBBA-EEFFDDE	15	2	7	11	CBAB-EEFFDED	IP + 3
10*	P12	P16	9	12	10	20	10	10	10	10	7	17	31	ACBB-FDE	20	7	17	31	ACBB-FDE	IP + 2
11	P12	P16	10	12	20	20	20	10	20	20	7	17	31	BCA-DEFFE	20	7	17	31	ABC-EFDFE	IP + 2
12[N]	P12	P16	8	16	10	10	10	5	5	5	5	12	22	BAC-DFE	20	5	12	22	BCA-FED	IP + 1
13	P16	P16	12	15	10	20	20	20	10	10	7	23	37	CABCB-EFDD	20	7	23	37	BCCBA-EDDF	IP + 2
14	P16	P16	16	14	10	40	20	40	20	20	7	20	34	CCBBABB-DEFD	20	7	20	34	CCBBABB-DEDF	IP + 1
15[N]	P16	P16	16	20	20	10	20	10	10	20	5	16	26	BAACC-FEDF	20	5	16	26	ACCBA-DFEF	IP + 0
16[N]	P16	K20	16	18	30	30	30	20	20	40	5	16	26	BCA-EFFD	20	5	16	26	BCA-FDEF	IP + 0
17[N]	P16	K20	18	18	10	10	10	5	15	10	4	14	22	CAB-EEDFEF	20	4	14	22	ACB-EEEDFF	IP + 2
18[N]	P16	K20	20	20	15	15	15	15	15	15	4	13	21	ABC-DFE	20	4	13	21	ACB-DEF	IP + 1
19*	P16	P24	14	16	40	20	20	40	20	10	6	22	34	ABAC-EFDEDDD	20	7	20	34	CAAB-EDFDDDE	IP + 3
20	P16	P24	16	18	15	45	30	20	40	20	5	17	27	BBCCBA-FEDE	20	5	17	27	CBCBBA-DEFE	IP + 2
21[N]	P16	P24	18	18	5	5	10	5	10	5	4	16	24	ABCC-EDFE	20	4	16	24	CABC-DFEE	IP + 3
22[N]	K20	K20	18	20	20	40	40	30	15	45	4	13	21	CABCB-FDFDEF	20	4	13	21	ABCBC-FDDFFE	IP + 2
23[N]	K20	K20	20	20	10	10	20	20	10	10	4	12	20	ABCC-DFDE	20	4	12	20	CBCA-EDDF	IP + 3
24[N]	K20	K20	22	22	10	5	10	10	10	5	3	12	18	CBAAC-EFEDD	20	3	12	18	CBAAC-EFEDD	IP + 1
25[N]	K20	P24	15	20	10	10	20	10	10	10	5	16	26	BACC-FED	20	5	16	26	BACC-DFE	IP + 2
26[N]	K20	P24	20	20	5	10	5	10	5	5	4	14	22	BABC-EFDD	20	4	14	22	BBCA-DFED	IP + 2

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Table 4. (Continued.)

#Test Case	Problem Data										Computational Results									
	Problem		Cycle Time		Demands L1			Demands L2			ABACO/S					ABACO/S-GA				
	Line I	*Line II*	*Line I*	*Line II*	*A*	*B*	*C*	*D*	*E*	*F*	*LL*	*NS*	*OBJ*	*Best Sequence Line I-Line II*	*NI*	*LL*	*NS*	*OBJ*	*Best Sequence Line I-Line II*	*NI*
27[N]	K20	P24	25	20	20	10	10	20	20	10	4	13	21	ACAB-FDDEE	20	4	13	21	ACBA-FEDED	IP + 3
28	P24	P24	15	20	20	10	10	10	10	10	5	19	29	ABCA-EFD	20	5	19	29	ABCA-EFD	IP + 4
29	P24	P24	25	20	10	20	10	10	20	20	4	14	22	CABB-EFEFD	20	4	14	22	BCBA-FEDEF	IP + 2
30[N]	P24	P24	18	24	10	10	20	5	10	15	4	15	23	CBCA-DEEFFF	20	4	15	23	CCAB-EFEFDF	IP + 1
31[N]	P24	K36	24	24	15	15	30	20	20	20	4	15	23	CABC-DFE	30	4	15	23	ABCC-DFE	IP + 1
32[N]	P24	K36	24	36	10	10	10	10	5	5	3	12	18	ACB-EDDF	30	3	12	18	ABC-DEDF	IP + 2
33[N]	P24	K36	18	36	5	10	5	4	2	4	4	14	22	CBAB-DEFDF	30	4	14	22	BABC-FEDDF	IP + 1
34[N]	K36	K36	20	30	5	5	20	5	5	10	5	18	28	CBACCC-FFED	30	5	18	28	CACBCC-EDFF	IP + 3
35[N]	K36	K36	36	36	5	5	5	5	5	5	3	12	18	CBA-EDF	30	3	12	18	CBA-FED	IP + 3
36[N]	K36	K36	40	30	10	10	10	20	10	10	4	12	20	BAC-DDFE	30	4	12	20	ABC-DFED	IP + 2
37	A65	A65	300	480	40	20	20	20	10	20	11	38	60	BACA-EFFDD	40	11	38	60	ABCA-DDEFF	IP + 2
38	A65	A65	420	360	15	15	30	20	40	10	10	37	57	CBCA-FDEEDEE	40	10	37	57	BCAC-DDFEEEE	IP + 2
39[N]	A65	A65	480	360	5	5	5	5	10	5	10	35	55	CAB-FEED	40	10	35	55	CBA-DEFE	IP + 2
40	A65	B148	405	810	10	5	5	4	4	2	9	32	50	AABC-EEDFD	40	9	32	50	BACA-EDFDE	IP + 3
41	A65	B148	675	540	20	10	10	10	20	20	9	31	49	ACBA-EFFED	40	9	31	49	BAAC-FFEDE	IP + 3
42[N]	A65	B148	630	720	20	10	10	10	10	15	7	28	42	ABCA-FFDEDFE	40	7	28	42	CABA-DFEFFDE	IP + 4
43	B148	B148	255	510	5	10	5	2	4	4	18	65	101	ABCB-EEFDF	40	18	65	101	ABBC-FEEFD	IP + 4
44	B148	B148	425	340	20	10	10	20	20	10	15	58	88	BACA-DEDEF	40	15	58	88	ABAC-EDEDF	IP + 3
45[N]	B148	B148	350	420	10	10	10	5	10	10	15	58	88	BCA-EFEDF	40	15	58	88	CBA-EFEDF	IP + 5

[N]: Newly generated test cases; *: Test cases modified by Kucukkoc and Zhang [36].

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

5.1. Stage 1

In this stage, the main objective of the computational tests is to investigate in how many iterations the proposed approach finds the same solution with Kucukkoc and Zhang [36]. For this aim, the algorithm is run for designated parameters and available problem data for each test case, and terminated when the same fitness value is found with Kucukkoc and Zhang [36]. Obtained results are reported in ABACO/S-GA column in Table 4 and ABACO/S column gives the results from Kucukkoc and Zhang [36] for the purpose of comparison. The newly generated test cases are marked with ^{N} superscript and solved using both ABACO/S and ABACO/S algorithms. The abbreviations *LL*, *NS*, and *OBJ* correspond to line length, number of workstations, and objective function value, respectively (where the user defined parameters in the objective function are considered as $\gamma_1 = 2$, and $\gamma_2 = 1$). *NI* columns exhibit the number of iterations that the ABACO/S and ABACO/S-GA algorithms are run.

Based on the coding structure of the ABACO/S, the number of sequences that the algorithm tried to find the best solution are considered as the number of iterations (in column *NI*) and are shown in the table. For the ABACO/S-GA, the number of iterations that the GA algorithm is run is shown in the *NI* column. In this column, '*IP + X*' means the best solution is found in the X^{th} iteration after generating the initial population. If *X* equals to zero, then it means that the best solution is found while generating the model chromosomes for the initial population. This issue (obtaining the best solution while generating the initial population) is observed for the majority of the small scale test cases.

As could be seen from the table, it is clear that the same objective values are obtained with the same *LL* and *NS* values except test case #19. In this test case, objective value of 34 is obtained with *LL=7*, and *NS=20*, different from the solution obtained by ABACO/S. Also, the same objective values are obtained with different model sequences except test cases #3, #10, #24 and #28.

According to the computational results, the proposed algorithm finds the same fitness values with the ABACO/S in less number of iterations. For instance, if we consider test case #2, ABACO/S-GA finds the objective value 15 in '*IP + 0'* iteration. In this case, 8 chromosomes are generated for the initial population and the same fitness value is found with the ABACO/S at this stage. For test case

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

#14, the proposed approach finds the objective value of 34 in '*IP + 1*' iterations, which means that 10 chromosomes are generated for the initial population and the best solution is discovered in the first iteration of crossover and mutation operations. One could argue that the total number of evaluated chromosomes depends on the number of offspring created after crossover and mutation operations, but the number of total chromosome evaluations are most likely less than the number of iterations considered by ABACO/S.

In terms of the performance of the proposed algorithm, the algorithm needs higher number of iterations for large-sized test cases compared with the small-sized test cases. However, this is caused by the exponentially growing search space with the increasing number of tasks.

5.2. Stage 2

In addition to the tests presented in Stage 1, ABACO/S-GA algorithm is run again for the same test cases (without the target objective function value) to observe whether the algorithm can find better solutions than the ABACO/S. ABACO/S-GA algorithm is run for 15, 20, 25 and 30 iterations for test cases #1-#9, #10-#30, #31-#36, and #37-#45, respectively. Among the solutions obtained, the ones better than ABACO/S are reported in Table 5 (please note that the same solutions have already been reported in Table 4 for the remaining problems).

Table 5. Computational results – Stage 2.

#Test Case	**ABACO/S**				**ABACO/S-GA**			
	LL	*NS*	*OBJ*	*Best Sequence Line I-Line II*	*LL*	*NS*	*OBJ*	*Best Sequence Line I-Line II*
16^N	5	16	26	BCA-EFFD	5	15	25	CBA-DEFF
25^N	5	16	26	BACC-FED	5	15	25	CABC-EDF
35^N	3	12	18	CBA-EDF	3	11	17	BCA-FED
36^N	4	12	20	BAC-DDFE	3	12	18	CAB-DDFE
38	10	37	57	CBCA-FDEEDEE	10	36	56	CABC-EFDDEEE
39^N	10	35	55	CAB-FEED	9	34	52	BAC-EDEF
42^N	7	28	42	ABCA-FFDEDFE	7	27	41	ACAB-FDEEDFF
43	18	65	101	ABCB-EEFDF	18	64	100	BCBA-EFDFE
44	15	58	88	BACA-DEDEF	15	57	87	BACA-DEEFD
45^N	15	58	88	BCA-EFEDF	15	56	86	BCA-FEDFE

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

As seen from the table, the results obtained using the proposed ABACO/S-GA algorithm for 10 test cases (i.e. #16, #25, #35, #36, #38, #39, #42 – #45) out of 45 test cases are lower than the solutions obtained by ABACO/S. The best improvement in the solutions obtained is observed for test case #39 for which ABACO/S-GA finds a solution with objective value of 52 while ABACO/S finds solution with objective value of 55.

There is no doubt that the solutions found by ABACO/S-GA are better than those obtained by ABACO/S for the same test cases. A Paired-samples t-Test is also conducted to prove this statistically. The effect of the integrated GA based engine on the objective function values of the obtained solutions is analyzed through a Paired-samples t-Test. The null and alternative hypotheses stated at the $a = 0.05$ level (95%) for means of objective function values obtained from ABACO/S (μ_{ABACOS}) and ABACO/S-GA ($\mu_{ABACOS-GA}$) are as follows:

H_0: The integrated GA based model sequencing mechanism has no significant positive effect on the objective function values of the obtained solutions ($\mu_{ABACOS} \leq \mu_{ABACOS-GA}$).

H_1: The integrated GA based model sequencing mechanism helps ABACO/S find solutions with better (reduced) objective function values ($\mu_{ABACOS} > \mu_{ABACOS-GA}$).

The input data used for the statistical test is given in Table 6 and the results of the test are reported in Table 7.

The test is one-tailed as it can be seen from the hypotheses. The results of the statistical analysis show that there is a significant difference between integrating ($\mu_{ABACOS-GA} = 30.4222, SD_{ABACOS-GA} = 20.2950$) and not integrating ($\mu_{ABACOS} = 30.7333, SD_{ABACOS} = 20.6270$) the GA based model sequencing procedure to the ABACO/S; $t(44) = 3.1234, p = 0.0016$. The results of the statistical test indicate that there is strong evidence to reject the null hypothesis and to argue that the integrated GA based procedure helps algorithm find solutions with better objective function values. Therefore, it is statistically proven that the integrated GA based procedure has significant effect on the objective function value of the obtained solution in a positive direction. In other words, ABACO/S finds better solutions for solved test cases of the MPTALB/S problem when a GA based model sequencing engine is integrated into it.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

Table 6. Data used for statistical test

Test Case	ABACO/S	ABACO/S-GA	Test Case	ABACO/S	ABACO/S-GA
1	16	16	24	18	18
2	15	15	25	26	25
3	11	11	26	22	22
4	14	14	27	21	21
5	12	12	28	29	29
6	12	12	29	22	22
7	20	20	30	23	23
8	16	16	31	23	23
9	11	11	32	18	18
10	31	31	33	22	22
11	31	31	34	28	28
12	22	22	35	18	17
13	37	37	36	20	18
14	34	34	37	60	60
15	26	26	38	57	56
16	26	25	39	55	52
17	22	22	40	50	50
18	21	21	41	49	49
19	34	34	42	42	41
20	27	27	43	101	100
21	24	24	44	88	87
22	21	21	45	88	86
23	20	20	-	-	-

Table 7. Paired Two-Sample t-Test for means of objective function values from ABACO/S and ABACO/S-GA

	ABACO/S	***ABACO/S GA***
Mean (μ)	30.7333	30.4222
Standard Deviation (SD)	20.6270	20.2950
Observations	45	45
Pearson Correlation	0.9995	
Hypothesized Mean Difference	0	
Degrees of Freedom (df)	44	
t Stat	**3.1234**	
$P(T \leq t)$ one-tail	**0.0016**	
t Critical one-tail	**1.6802**	
$P(T \leq t)$ two-tail	0.0032	
t Critical two-tail	2.0154	

6. Conclusions

A GA based model sequencing procedure is developed and integrated to the agent based ant colony optimization based approach for solving the problem of simultaneous balancing and sequencing of

mixed-model parallel two-sided assembly lines effectively. The main aim is to improve the competence of model sequencing procedure of the entire algorithm to robustly find better quality solutions than the previously developed agent based ACO approach for the same instances. This paper contributes to the scientific knowledge in terms of its novel methodology which incorporates GA and agent based ACO algorithm to solve one of the recently introduced and rarely studied manufacturing engineering problems. Moreover, to the best knowledge of the authors, this is the first hybridization of GA and ACO algorithms to balance any type of parallel assembly lines in the literature.

An example is given to illustrate the running mechanism of the proposed technique and evolution of the chromosomes through generations. To assess the performance of the developed technique, test cases are solved and the iteration numbers that the best solutions obtained are compared with the results available in the literature. When the algorithm is run one more time for the same test cases without terminating when the specific objective value is found, better solutions are achieved for 10 of the test cases than those existing in the literature. In addition to the test cases available in the literature, new test cases are build using newly generated test problems and their combinations with the existing test problems, and are used as input to both ABACO/S and ABACO/S-GA algorithms. A Paired-samples t-Test is conducted to statistically analyze the performance of the newly developed algorithm. Test results statistically prove that the developed algorithm has a better solution capacity than the existing method available in the literature in terms of solving the provided test cases.

In the literature, demand fluctuations, resource dependent task times and workforce efficiency have been introduced to relatively simple assembly line balancing types, *i.e.* simple straight one-sided lines with no parallelization and/or model variations. Also, model sequencing problem has not been considered in such studies. As we are dealing with a new as well as complex problem type, model demands are assumed deterministic and operators are considered multi-skilled, *i.e.* they have preferences neither on tasks nor on workstations. Relaxation of some of these assumptions in future studies may lead to more interesting as well as complex problem types to cope with. Also, processing times of tasks may also depend on the workstations in which they are performed. All of these developments are of big research issues which can expand the current study and put it one step

forward. Some other new meta-heuristics could also be developed to solve the MPTALB/S problem in future studies. New procedures could be hybridized with the agent based ACO algorithm and their performances could be compared with the proposed ABACO/S-GA approach presented in this research.

Appendices

Figure A1: Precedence relationships diagram for test problem K20

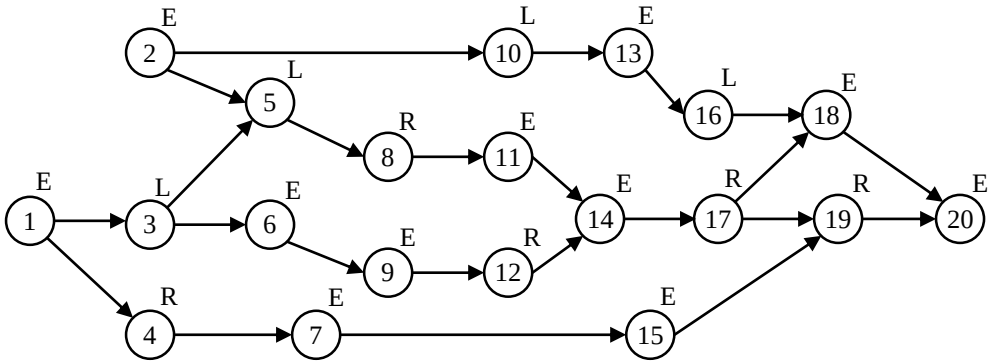

Figure A2: Precedence relationships diagram for test problem K36

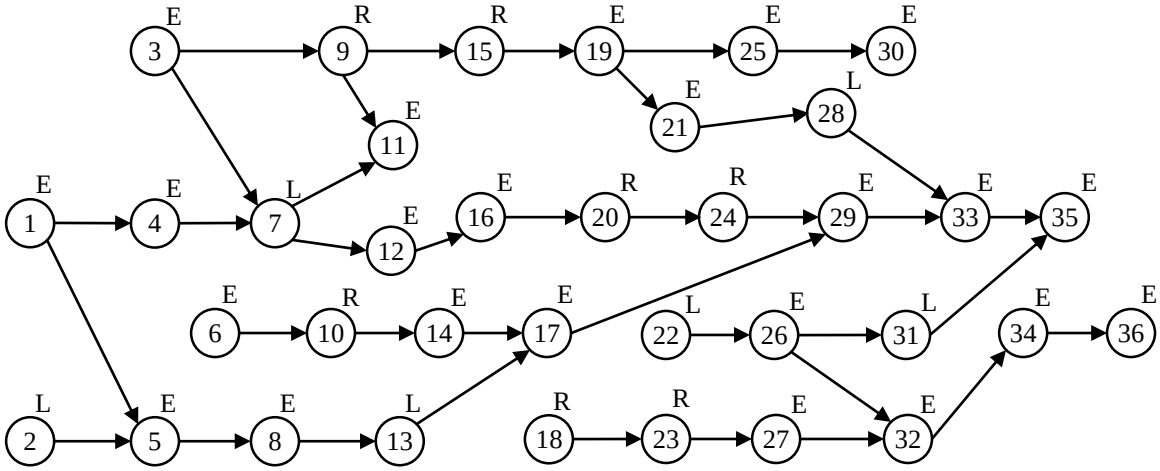

Table A1: Processing times of tasks belonging to K20 and K36

Task Number	K20			K36		
	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
1	4	7	3	9	5	4
2	3	5	9	5	3	9
3	0	2	3	7	7	9
4	4	1	3	2	3	0
5	1	2	2	6	4	5
6	4	8	1	4	8	9
7	3	4	9	4	1	4
8	5	4	5	1	4	1
9	7	7	6	0	6	2
10	8	3	0	8	5	3
11	2	6	3	0	8	0
12	1	6	5	6	5	0
13	2	1	9	7	6	7
14	5	8	9	1	1	1
15	3	5	3	5	9	1
16	2	1	4	6	5	7
17	5	2	5	0	0	0
18	0	8	6	7	3	1
19	2	0	1	8	3	5
20	4	0	9	2	2	3
21	-	-	-	8	0	8
22	-	-	-	1	3	8
23	-	-	-	1	2	0
24	-	-	-	7	1	8
25	-	-	-	5	1	3
26	-	-	-	5	0	6
27	-	-	-	9	0	0
28	-	-	-	1	3	5
29	-	-	-	2	6	8
30	-	-	-	5	0	5
31	-	-	-	5	1	4
32	-	-	-	0	5	1
33	-	-	-	7	8	1
34	-	-	-	7	4	7
35	-	-	-	8	7	3
36	-	-	-	8	1	2

References

1. Ozdemir RG, Ayag Z (2011) An integrated approach to evaluating assembly-line design alternatives with equipment selection. Production Planning & Control 22 (2):194-206.
2. Battini D, Faccio M, Persona A, Sgarbossa F (2009) Balancing-sequencing procedure for a mixed model assembly system in case of finite buffer capacity. International Journal of Advanced Manufacturing Technology 44 (3-4):345-359.
3. Boysen N, Fliedner M, Scholl A (2009) Assembly line balancing: Joint precedence graphs under high product variety. Iie Transactions 41 (3):183-193.
4. Thomopoulos NT (1967) Line Balancing-Sequencing for Mixed-Model Assembly. Management Science 14 (2):59-75.
5. Akpinar S, Bayhan GM, 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.
6. Kucukkoc I, Zhang DZ (2014) Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Research 52 (12):3665-3687.

The final publication is available at: http://link.springer.com/article/10.1007/s00170-015-7320-y

7. Erel E, Gokcen H (1999) Shortest-route formulation of mixed-model assembly line balancing problem. European Journal of Operational Research 116 (1):194-204.
8. Matanachai S, Yano CA (2001) Balancing mixed-model assembly lines to reduce work overload. Iie Transactions 33 (1):29-42.
9. Vilarinho PM, Simaria AS (2002) A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. International Journal of Production Research 40 (6):1405-1420.
10. McMullen PR, Tarasewich P (2003) Using ant techniques to solve the assembly line balancing problem. Iie Transactions 35 (7):605-617.
11. Yagmahan B (2011) Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. Expert Systems with Applications 38 (10):12453-12461.
12. Hamta N, Ghomi SMTF, Jolai F, Shirazi MA (2013) A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. International Journal of Production Economics 141 (1):99-111.
13. Kucukkoc I, Karaoglan AD, 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.
14. Yano CA, Rachamadugu R (1991) Sequencing to Minimize Work Overload in Assembly Lines with Product Options. Management Science 37 (5):572-586.
15. Kim YK, Hyun CJ, Kim Y (1996) Sequencing in mixed model assembly lines: A genetic algorithm approach. Computers & Operations Research 23 (12):1131-1145.
16. Zheng YQ, Wang YP, Hu B, Wang YS (2011) A sequencing approach of models in mixed-model assembly lines. Mechanika 17 (4):418-422.
17. Bautista J, Cano A (2011) Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules. European Journal of Operational Research 210 (3):495-513.
18. Zhu XW, Hu SJ, Koren Y, Huang NJ (2012) A complexity model for sequence planning in mixed-model assembly lines. Journal of Manufacturing Systems 31 (2):121-130.
19. Manavizadeh N, Tavakoli L, Rabbani M, Jolai F (2013) A multi-objective mixed-model assembly line sequencing problem in order to minimize total costs in a Make-To-Order environment, considering order priority. Journal of Manufacturing Systems 32 (1):124-137.
20. Xu S, Li FM (2013) A Modified Genetic Algorithm for Sequencing Problem in Mixed Model Assembly Lines. Advanced Materials Research 655-657 (2013):1675-1681.
21. Boysen N, Fliedner M, Scholl A (2009) Sequencing mixed-model assembly lines: Survey, classification and model critique. European Journal of Operational Research 192 (2):349-373.
22. Kucukkoc I, Zhang DZ (2015) A Mathematical Model and Genetic Algorithm based Approach for Parallel Two-Sided Assembly Line Balancing Problem. Production Planning & Control doi: 10.1080/09537287.2014.994685.
23. Bartholdi JJ (1993) Balancing 2-Sided Assembly Lines - a Case-Study. International Journal of Production Research 31 (10):2447-2461.
24. Kim YK, Kim SJ, Kim JY (2000) Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm. Production Planning & Control 11 (8):754-764.
25. Gökçen H, Agpak K, Benzer R (2006) Balancing of parallel assembly lines. International Journal of Production Economics 103 (2):600-609.
26. Ozcan U, Gokcen H, Toklu B (2010) Balancing parallel two-sided assembly lines. International Journal of Production Research 48 (16):4767-4784.
27. Kucukkoc I, Zhang DZ (2013) Balancing Parallel Two-Sided Assembly Lines via a Genetic Algorithm Based Approach. Paper presented at the Proceedings of the 43rd International Conference on Computers and Industrial Engineering (CIE43), The University of Hong Kong, Hong Kong, October 16-18

28. Kucukkoc I, Zhang DZ (2015) Type-E Parallel Two-Sided Assembly Line Balancing Problem: Mathematical Model and Ant Colony Optimisation based Approach with Optimised Parameters. Computers & Industrial Engineering doi: 10.1016/j.cie.2014.12.037.
29. Simaria AS, Vilarinho PM (2009) 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. Computers & Industrial Engineering 56 (2):489-506.
30. Ozcan U, Toklu B (2009) Balancing of mixed-model two-sided assembly lines. Computers & Industrial Engineering 57 (1):217-227.
31. 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.
32. Ozcan U, Cercioglu H, Gokcen H, Toklu B (2010) Balancing and sequencing of parallel mixed-model assembly lines. International Journal of Production Research 48 (17):5089-5113.
33. Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics 142 (2):259-277.
34. Zhang DZ, Kucukkoc I (2013) Balancing Mixed-Model Parallel Two-Sided Assembly Lines. Paper presented at the Proceedings of the International Conference on Industrial Engineering and Systems Management (IEEE-IESM'2013), Rabat, Morocco, October 28 - 30
35. Kucukkoc I, Zhang DZ (2014) An Agent Based Ant Colony Optimisation Approach for Mixed-Model Parallel Two-Sided Assembly Line Balancing Problem. Paper presented at the Pre-Prints of the Eighteenth International Working Seminar on Production Economics, Innsbruck, Austria, 24-28 February
36. Kucukkoc I, Zhang DZ (2014) 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.
37. Chryssolouris G (2006) Manufacturing Systems: Theory and Practice (2nd Edition). SpringerVerlag, New York
38. Song BL, Wong WK, Fan JT, Chan SF (2006) A recursive operator allocation approach for assembly line-balancing optimization problem with the consideration of operator efficiency. Computers & Industrial Engineering 51 (4):585-608.
39. Corominas A, Pastor R, Plans J (2008) Balancing assembly line with skilled and unskilled workers. Omega-International Journal of Management Science 36 (6):1126-1132.
40. Manavizadeh N, Hosseini NS, Rabbani M, Jolai F (2013) A Simulated Annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach. Computers & Industrial Engineering 64 (2):669-685.
41. Koltai T, Tatay V (2013) Formulation of workforce skill constraints in assembly line balancing models. Optimization and Engineering 14 (4):529-545.
42. 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.
43. Jayaswal S, Agarwal P (2014) Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach. Journal of Manufacturing Systems 33 (4):522-534.
44. Kara Y, Özgüven C, Yalçın N, Atasagun Y (2011) Balancing straight and U-shaped assembly lines with resource dependent task times. International Journal of Production Research 49 (21):6387-6405.
45. Gajpal Y, Rajendran C, Ziegler H (2006) An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. The International Journal of Advanced Manufacturing Technology 30:416-424.
46. Bard JF, Darel E, Shtub A (1992) An Analytic Framework for Sequencing Mixed Model Assembly Lines. International Journal of Production Research 30 (1):35-48.

47. Costa A, Cappadonna FA, Fichera S (2014) Joint optimization of a flow-shop group scheduling with sequence dependent set-up times and skilled workforce assignment. International Journal of Production Research doi: 10.1080/00207543.2014.883469.
48. Ugarte BS, Pellerin R, Artiba A (2011) An improved genetic algorithm approach for on-line optimisation problems. Production Planning & Control 22 (8):742-753.
49. 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.
50. Holland JH (1975) Adaptation in Natural and Artificial Systems. MIT Press, Cambridge
51. Anosike AI, Zhang DZ (2009) An agent-based approach for integrating manufacturing operations. International Journal of Production Economics 121 (2):333-352.
52. Liang H, Su H, Wang X, Chen MZQ (2014) Swarm aggregations of heterogeneous multi-agent systems. International Journal of Control 87 (12):2594-2603.
53. Cordon O, Herrera F, Stützle T (2002) A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends Mathware & Soft Computing 9 (3):141-175.
54. Blum C (2005) Ant colony optimization: Introduction and recent trends. Physics of Life Reviews 2 (4):353-373.
55. Srinivas M, Patnaik LM (1994) Genetic Algorithms - a Survey. Computer 27 (6):17-26.
56. Li YC, Zhao LN, Zhou SJ (2011) Review of Genetic Algorithm. Advanced Materials Research 179-180:365-367.
57. Leitao P (2009) Agent-based distributed manufacturing control: A state-of-the-art survey. Engineering Applications of Artificial Intelligence 22 (7):979-991.
58. Tasan SO, Tunali S (2008) A review of the current applications of genetic algorithms in assembly line balancing. Journal of Intelligent Manufacturing 19 (1):49-69.
59. Lee ZJ, Su SF, Chuang CC, Liu KH (2008) Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Applied Soft Computing 8 (1):55-78.
60. Chen SM, Chien CY (2011) Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Systems with Applications 38 (4):3873-3883.
61. Li N, Wang S, Li Y (2011) A Hybrid Approach of GA and ACO for VRP. Journal of Computational Information Systems 7 (13):4939-4946.
62. Arcus A (1966) COMSOAL, A Computer Method of Sequencing Operations for Assembly Lines. The International Journal of Production Research 4 (4):259-277.
63. Helgeson WB, Birnie DP (1961) Assembly Line Balancing Using the Ranked Positional Weight Technique. The Journal of Industrial Engineering 12 (6):394-398.
64. Talbot FB, Patterson JH (1984) An Integer Programming Algorithm with Network Cuts for Solving the Assembly Line Balancing Problem. Management Science 30 (1):85-99.
65. 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.
66. Arcus AL (1963) An analysis of a computer method of sequencing assembly line operations. Ph.D. Dissertation, University of California, Berkeley, USA,
67. Tonge FM (1960) Summary of a Heuristic Line Balancing Procedure. Management Science 7 (1):21-42.
68. Kim YK, Kim YH, Kim YJ (2000) Two-sided assembly line balancing: a genetic algorithm approach. Production Planning & Control 11 (1):44-53.
69. Lee TO, Kim Y, Kim YK (2001) Two-sided assembly line balancing to maximize work relatedness and slackness. Computers & Industrial Engineering 40 (3):273-292.