Journal of Intelligent Manufacturing

A mathematical model and artificial bee colony algorithm for the lexicographic bottleneck mixed-model assembly line balancing problem

Ibrahim Kucukkoc [ab*], Kadir Buyukozkan [cd], Sule Itir Satoglu [c], David Z. Zhang [a]

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

[b] Department of Industrial Engineering, Faculty of Engineering and Architecture, Balikesir University, Cagis Campus, Balikesir, Turkey

[c] Industrial Engineering Department, Istanbul Technical University, Istanbul, Turkey

[d] Department of Industrial Engineering, Faculty of Engineering, Karadeniz Technical University, Kanuni Campus, Trabzon, Turkey

Abstract

Typically, the total number of required workstations are minimised for a given cycle time (this problem is referred to as type-1), or cycle time is minimised for a given number of workstations (this problem is referred to as type-2) in traditional balancing of assembly lines. However, variation in workload distributions of workstations is an important indicator of the quality of the obtained line balance. This needs to be taken into account to improve the reliability of an assembly line against unforeseeable circumstances, such as breakdowns or other failures. For this aim, a new problem, called lexicographic bottleneck mixed-model assembly line balancing problem (LB-MALBP), is presented and formalised. The lexicographic bottleneck objective, which was recently proposed for the simple single-model assembly line system in the literature, is considered for a mixed-model assembly line system. The mathematical model of the LB-MALBP is developed for the first time in the literature and coded in GAMS solver, and optimal solutions are presented for some small scale test problems available in the literature. As it is not possible to get optimal solutions for the large-scale instances, an artificial bee colony algorithm is also implemented for the solution of the LB-MALBP. Solution procedures of the algorithm are explored illustratively. The performance of the algorithm is also assessed using derived well-known test problems in this domain and promising results are observed in reasonable CPU times.

Keywords: lexicographic bottleneck; assembly line balancing; mixed-model lines; mathematical model; artificial bee colony algorithm.

* Corresponding author: Ibrahim Kucukkoc,

Email: i.kucukkoc@exeter.ac.uk, ikucukkoc@balikesir.edu.tr Tel: +441392723613.

K.B Email: kbuyukozkan@ktu.edu.tr, Tel: +904623772954; S.I.S Email: onbaslis@itu.edu.tr, Tel: +902122856801; D.Z.Z Email: d.z.zhang@exeter.ac.uk, Tel: +441392723641.

Journal of Intelligent Manufacturing

A mathematical model and artificial bee colony algorithm for the lexicographic bottleneck mixed-model assembly line balancing problem

Abstract

Typically, the total number of required workstations are minimised for a given cycle time (this problem is referred to as type-1), or cycle time is minimised for a given number of workstations (this problem is referred to as type-2) in traditional balancing of assembly lines. However, variation in workload distributions of workstations is an important indicator of the quality of the obtained line balance. This needs to be taken into account to improve the reliability of an assembly line against unforeseeable circumstances, such as breakdowns or other failures. For this aim, a new problem, called lexicographic bottleneck mixed-model assembly line balancing problem (LB-MALBP), is presented and formalised. The lexicographic bottleneck objective, which was recently proposed for the simple single-model assembly line system in the literature, is considered for a mixed-model assembly line system. The mathematical model of the LB-MALBP is developed for the first time in the literature and coded in GAMS solver, and optimal solutions are presented for some small scale test problems available in the literature. As it is not possible to get optimal solutions for the large-scale instances, an artificial bee colony algorithm is also implemented for the solution of the LB-MALBP. Solution procedures of the algorithm are explored illustratively. The performance of the algorithm is also assessed using derived well-known test problems in this domain and promising results are observed in reasonable CPU times.

Keywords: lexicographic bottleneck; assembly line balancing; mixed-model lines; mathematical model; artificial bee colony algorithm.

1. Introduction

An *assembly line* is a manufacturing process composed of the sequence of *workstations* in which parts are added to a semi-finished product as it moves from workstation to workstation with the help of a transportation system such as a conveyor or moving belt. Semi-finished products pass workstations in which *operations* (or *tasks*) required to create an end product are handled by operators or machines in a sequential manner (Kucukkoc and Zhang 2015a; Satoglu and Sahin 2012). The *assembly line* *balancing problem* is the effort of determining which task will be performed in which workstation, where each task requires a certain amount of time to be completed, called *task time* or *task processing time*. The sum of processing times of tasks assigned to a workstation cannot exceed the capacity of that workstation, designated by the *cycle time* (which also determines the *production rate* or *throughput rate*). Due to some technological or organisational restrictions, some tasks need to be completed before initialising some other tasks. This restriction is called *precedence relationship constraint* and must be satisfied for all tasks to obtain feasible balancing solutions. Another essential constraint is that tasks cannot be split between workstations, which means every task must be assigned to exactly one workstation.

Assembly lines can be classified as *single-model lines*, *mixed-model lines* and *multi-model lines*. Only one model of a product is assembled on a single-model line while more than one similar model of a base product is assembled on a mixed-model line in an intermixed order. As the models being assembled on a mixed-model line are similar to each other, there is no setting-up process needed between model changes. Nevertheless, the set-up is needed on multi-model lines, on which considerably different product models are assembled.

In traditional approach, the main objective of balancing an assembly line is to maximise the efficiency of the line either by minimising the number of workstations (this problem is referred to as type-I) or minimising the cycle time (this problem is referred to as type-II). In type-I problems, it is aimed to minimise the number of workstations, which is thought to be the master piece of the cost of establishing and running an assembly line, while the cycle time is determined and known beforehand. These type of problems are usually dealt with when building a new line. On the other hand, the type-II problems tackle minimising the cycle time when the number of workstations is given. These type of problems occur usually when rebalancing of a line due to changes in product demand or the capacity of the line. Type-II problems adopt min-max objectives since it is aimed to minimise maximum workload of the most heavily loaded workstation (called bottleneck), while workloads of other workstations are often ignored. However, as it is emphasised by Boysen *et al*. (2007), considering the second biggest, third biggest, *etc.* workloads is also important to prevent the quality defects caused by disproportionately distributed workloads. Thus, the reliability of the system is improved and the workload is likely to be distributed among workers as equally as possible.

From that base point, Pastor (2011) proposed a new assembly line balancing problem, namely the *lexicographic bottleneck assembly line balancing problem (LB-ALBP)*, which aimed at hierarchically minimising the workload of the most heavily loaded workstation, and the workload of the second most heavily loaded workstation, and so on for simple assembly lines. Two mixed-integer linear programming models were designed to solve the LB-ALBP optimally, together with three heuristic procedures based on these models. Pastor *et al*. (2012) proposed and tested new algorithms which were different combinations of a heuristic procedure and several local search procedures derived from different line balancing procedures. Recently, Pastor *et al.* (2015) employed GRASP, scatter search and tabu search for solving the LB-ALBP and compared their performance to each other, on the basis of well-known test problems. The authors reported that the GRASP and scatter search integrated with an improvement method reached better solutions than the best heuristic that was proposed by Pastor *et al.* (2012).

The basic type-I and type-II problems have been studied extensively in the literature and various exact, heuristic and meta-heuristic solution approaches have been developed in the literature so far. See for example, Baykasoglu and Dereli (2009), Simaria and Vilarinho (2009), Kara and Tekin (2009), Yagmahan (2011), Xu and Xiao (2011), Chutima and Chimklai (2012), Hamzadayi and Yildiz (2012), Rabbani *et al*. (2012), Liao *et al*. (2012), Akpinar *et al*. (2013), Kucukkoc *et al*. (2013), Manavizadeh *et al*. (2013) and Kucukkoc and Zhang (2015a, 2015b, 2014a) for recently published studies on type-I problems; and Hackman *et al*. (1989), Simaria and Vilarinho (2004), Liu *et al*. (2004), Battini *et al*. (2007), Ozcan *et al*. (2011) and Yoosefelahi *et al*. (2012) for studies published on type-II problems. While these two objectives conflict each other, both cycle time and the number of workstations are minimised in some studies (*e.g.* Wei and Chao (2011), García-Villoria and Pastor (2013), Manavizadeh *et al*. (2012) and Kucukkoc and Zhang (2015c)). Also, various types of assembly line balancing problems (including consideration of U-shaped lines, parallel lines, two-sided lines, stochastic task times, buffers in between workstations, *etc*.) have been dealt with by extensive numbers of researchers.

Two recent studies by Battaïa and Dolgui (2013) and Sivasankaran and Shahabudeen (2014) provided a comprehensive review of the literature and presented a taxonomy of line balancing problems and their solution approaches. The reader may also refer to Baybars (1986), Ghosh and Gagnon (1989), Rekiek *et al*. (2002), Becker and Scholl (2006), and Boysen *et al*. (2007).

Ever since the mixed-model line balancing problem was first introduced by Thomopoulos (1967), several exact, heuristic and meta-heuristic methods were proposed for the solution of the problem with different objectives, see for example Askin and Zhou (1997), Gokcen and Erel (1997), Vilarinho and Simaria (2002), McMullen and Tarasewich (2003), Haq *et al.* (2006), Kara *et al.* (2007a, 2007b), Ozcan and Toklu (2009), Hwang and Katayama (2009, 2010), Emde *et al*. (2010), Akgunduz and Tunali (2010), Zhang and Gen (2011), Chutima and Chimklai (2012), Rabbani *et al.* (2012), Mosadegh *et al.* (2012), Liao *et al.* (2012), Zhang and Kucukkoc (2013), Hamta *et al.* (2013), Manavizadeh *et al.* (2013) and Kucukkoc and Zhang (2014b; 2015d).

Although variation of task processing times between different models in mixed-model lines is likely to cause several problems (as explained above), the lexicographic bottleneck objective has not been studied for any type of mixed-model assembly line balancing problem so far. In other words, the LB-ALBP, which is different from type-2 line balancing problem as it was exposed by Pastor (2011) and Pastor *et al*. (2012), has never been studied for a mixed-model line in the literature. Based on this motivation, this paper is original in terms of both the addressed problem along with its mathematical formulation and the proposed solution method. We define the lexicographic bottleneck mixed-model assembly line balancing problem (LB-MALBP) mathematically, the primitive version of which was introduced by Pastor (2011) for the simple assembly line balancing problem. Also, in this paper, an artificial bee colony algorithm is applied for any type of LB-ALBP for the first time in the literature.

The remainder of the paper is organised as follows. Section 2 defines the LB-MALBP mathematically while Section 3 presents the proposed solution method and gives optimal and heuristic solutions of a numerical example. An experimental study is conducted in Section 4 and detailed solutions are presented for twenty test problems derived from the literature. Conclusions are drawn along with practical implications and future research directions in Section 5.

2. The lexicographic bottleneck mixed-model assembly line balancing problem

2.1. Definition and main characteristics

The mixed-model assembly line is composed of sequential workstations, represented with $(k = 1, \dots, K)$, in which various models of a base product are assembled in an intermixed order. Each model, symbolised with $(m = 1, \dots, P)$, has its own set of tasks, where a task is represented by i $(i = 1, \dots, N)$. The processing time of a task, represented with t_{im}, can be different for different product models while the sum of processing times of tasks assigned to a workstation cannot exceed the upper limit designated for cycle time, CT, for any product model. A common (or joint) precedence relationships diagram is built and thus common tasks between different models are assigned to the same workstation. The precedence relationships between task pairs must be ensured to obtain feasible balancing solutions where O holds the set of ordered pairs of tasks (i, j) such that there is an immediate precedence relation between them, *i.e.* task i is an immediate predecessor of task j.

Due to the differences in task processing times between different product models, fluctuations in workload distributions of workstations are more likely to occur in mixed-model lines rather than in single-model lines. Also, it is possible to have different balancing solutions which have the same traditional performance measures, such as the total number of workstations or cycle time. To give an example, let us consider an example problem for which the precedence relationships diagram and task processing times are presented in Figure 1 and Table 1, respectively. As seen from the data, three different models (A, B and C) are considered. Model demands are assumed 16, 24 and 8 units for models A, B, and C respectively ($D_A = 16$, $D_B = 24$, $D_C = 8$) for a planning period of 480 time units.

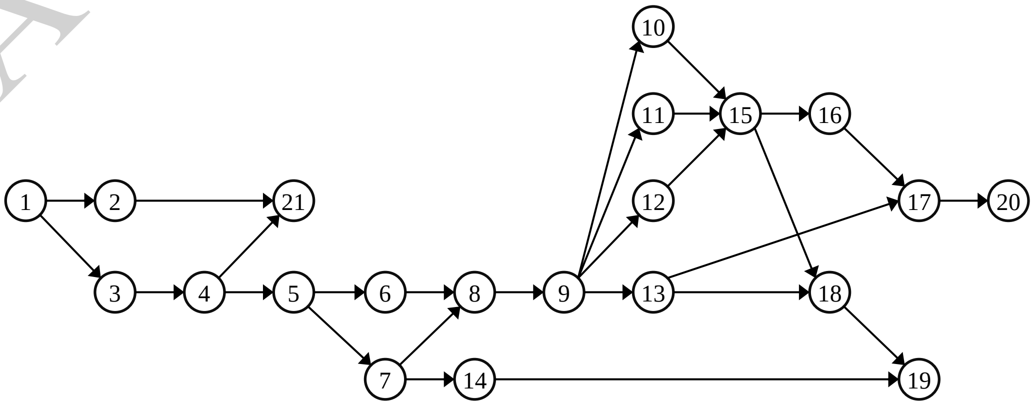


Figure 1. Combined precedence relationships diagram of the example problem

Table 1 . Task processing times of the models

Task No	Model A	Model B	Model C	Task No	Model A	Model B	Model C
1	5.90	5.90	5.90	12	0.00	1.00	1.00
2	6.05	6.05	6.05	13	0.00	5.00	5.00
3	7.00	7.00	7.00	14	2.70	2.70	2.40
4	5.40	5.00	5.50	15	7.50	7.50	7.50
5	6.50	7.20	6.50	16	2.40	0.00	2.40
6	3.20	0.00	3.20	17	2.10	2.10	1.80
7	5.00	5.80	5.20	18	2.80	2.80	2.80
8	6.30	6.70	7.20	19	7.30	7.30	7.30
9	5.00	5.00	5.00	20	5.60	4.80	5.60
10	0.00	0.00	10.00	21	0.00	1.00	1.00
11	4.50	5.15	0.00	-	-	-	-

The cycle time is computed as $CT = H/\sum_{m=1}^{P} D_m = 480/48 = 10$ for this example problem and two possible balancing solutions are provided in Table 2 and Table 3. Weighted workloads of the workstations (S_k), which are calculated using Equation (1) based on the model demands, are also given as well as individual workload times in model basis.

$$S_k = \sum_{m=1}^{P} d_m \left(\sum_{i \in AT_k} t_{im} \right), \tag{1}$$

where AT_k and t_{im} denote the set of tasks assigned to workstation k and the processing time of task i for model m, respectively. Please note that d_m corresponds to proportional demand for model m and is calculated as follows, $d_m = D_m/\sum_{m=1}^{P} D_m$, where D_m is the demand of model m.

The detailed balancing solution results provided in Table 2 and Table 3 are also plotted in Figure 2 to observe fluctuations in workload distributions of workstations in accordance with different model types (see A, B, and C bars in the figure). Also, the weighted workload of each workstation is given as a stacked line to make comparison easier between alternative balancing solutions.

It is apparent that the provided two alternative solutions are quite different from each other and should have different effects on the reliability and criticalness of the line, where the closer station workload to cycle time the more critical obtained line configuration. The lexicographic bottleneck objective proposed in this research aims to minimise the most heavily loaded workstation, followed by the second most heavily loaded workstation, and so on, on the mixed-model assembly lines. Thus, a sophisticated study is to be maintained to obtain smoother workload distributions in the established line balancing solutions. The next subsection provides the mathematical formulation of the LB-MALBP for the first time in the literature.

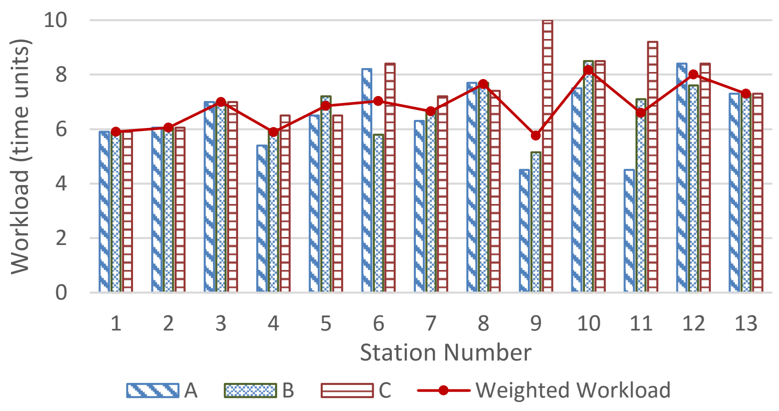
Table 2. Balancing solution - 1

Station No	Assigned Tasks	Weighted Workload	Model A	Model B	Model C
1	1	5.900	5.90	5.90	5.90
2	2	6.050	6.05	6.05	6.05
3	3	7.000	7.00	7.00	7.00
4	4, 21	5.887	5.40	6.00	6.50
5	5	6.850	6.50	7.20	6.50
6	7, 6	7.034	8.20	5.80	8.40
7	8	6.653	6.30	6.70	7.20
8	9, 14	7.649	7.70	7.70	7.40
9	10, 11	5.760	4.50	5.15	10.00
10	12, 15	8.170	7.50	8.50	8.50
11	13, 16, 17	6.599	4.50	7.10	9.20
12	20, 18	8.000	8.40	7.60	8.40
13	19	7.300	7.30	7.30	7.30

Table 3. Balancing solution - 2

Station No	Assigned Tasks	Weighted Workload	Model A	Model B	Model C
1	1	5.900	5.90	5.90	5.90
2	3	7.000	7.00	7.00	7.00
3	4	5.217	5.40	5.00	5.50
4	5	6.850	6.50	7.20	6.50
5	7, 6	7.034	8.20	5.80	8.40
6	8	6.653	6.30	6.70	7.20
7	9, 12	5.670	5.00	6.00	6.00
8	10, 11	5.760	4.50	5.15	10.00
9	15	7.500	7.50	7.50	7.50
10	2, 16, 21	7.920	8.45	7.05	9.45
11	13, 17, 14	8.048	4.80	9.80	9.20
12	20, 18	8.000	8.40	7.60	8.40
13	19	7.300	7.30	7.30	7.30

(a)



(b)

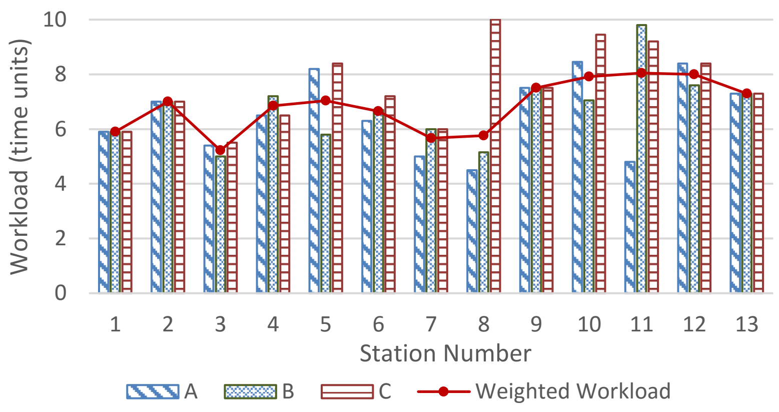


Figure 2. Workload distribution among workstations: (a) solution-1, (b) solution-2

2.2. Mathematical model

This subsection presents the developed mathematical model of LB-MALBP, which is an improved version of the formulation provided by Pastor (2011) for single-model assembly lines.

2.2.1. Notation

m	index of models
i, j	index of tasks
k, l, n	index of workstations
N	the total number of tasks $(i = 1, \ldots, N)$
K	the total number of workstations $(k = 1, \ldots, K)$
P	the total number of models $(m = 1, \ldots, P)$
t_{im}	the processing time of task i for model m

H the planning horizon

D_m the demand of model m in the planning horizon

d_m the proportional demand of model m, where $d_m = D_m / \sum_{m=1}^{P} D_m$

CT the upper bound on the cycle time, where $CT = H / \sum_{m=1}^{P} D_m$

O the set of ordered pairs of tasks (i, j) such that there is an immediate precedence relation between them, task i is an immediate predecessor of task j

α_k parameters to weigh the components of the objective function $(k = 1, \dots, K)$. So that $\alpha_k \gg \alpha_{k+1}$ guarantees that the hierarchy of the proposed objectives is preserved.

2.2.2. Variables

$$v_{ik} = \begin{cases} 1 & \mathit{if\ and\ only\ if\ task\ i\ is\ assigned\ to\ workstation\ k} \\ 0 & \mathit{otherwise} \end{cases}$$

S_k weighted workload assigned to workstation k (*i.e.*, the sum of the weighted processing times of the tasks assigned to the workstation k)

T_k the weighted workload of the k-th most heavily loaded workstation. T_1 is the weighted workload of the most heavily loaded workstation; T_2 is the weighted workload of the second most heavily loaded station, *etc.*

$$y_{kn} = \begin{cases} 1 & \mathit{if\ and\ only\ if\ station\ k\ has\ the\ n-th\ highest\ weighted\ workload}, T_n \\ 0 & \mathit{otherwise} \end{cases}.$$

2.2.3. Objective function

$$min\ z = \sum_{k=1}^{m} \alpha_k \cdot T_k \tag{2}$$

2.2.4. Constraints

$$\sum_{k=1}^{K} v_{ik} = 1, \quad \forall i \tag{3}$$

$$\sum_{m=1}^{P} \sum_{i=1}^{N} (d_m \cdot t_{im} \cdot v_{ik}) = S_k, \quad \forall k \tag{4}$$

$$\sum_{i=1}^{N}(t_{im} \cdot v_{ik}) \leq CT, \quad \forall k, \forall m \tag{5}$$

$$\sum_{k=1}^{K} y_{kn} = 1, \quad \forall n \tag{6}$$

$$\sum_{n=1}^{K} y_{kn} = 1, \quad \forall k \tag{7}$$

$$\sum_{k=1}^{K} k \cdot \left(v_{ik} - v_{jk}\right) \leq 0, \quad \forall (i,j) \in O \tag{8}$$

$$S_k \leq T_n + \sum_{l=1}^{n-1}(CT \cdot y_{kl}), \quad \forall k, \forall n \tag{9}$$

$$\sum_{k=1}^{K} T_k \leq \sum_{n=1}^{K} S_n \tag{10}$$

$$S_k, T_k \geq 0 \tag{11}$$

The objective function (2) minimises the weighted sum of the weighted workloads of the workstations with taking into account the hierarchy of proposed objectives. Constraint (3) implies that every task is assigned to one and only one workstation. Constraint (4) calculates the weighted processing time (weighted workload) assigned to each workstation. Constraint (5) assures that the workload of the workstation for any model cannot be greater than the upper bound of cycle time. Constraint (6) imposes that each workstation is assigned to only one position in the list of workstations (ordered by workload). Constraint (7) imposes that each position of the workstation list, ordered by workload, is assigned to only one workstation. Constraint (8) ensures that the technological precedence conditions are not violated. Constraint (9) imposes that if workstation k is the n-th most heavily loaded workstation, its weighted workload will not be greater than the weighted workload of the first n most heavily loaded workstations. Constraint (10) assures that $T_k$variable gets the smallest value among the possible values in the range. Constraint (11) defines signs for variables.

3. Solution methods

Two different solution approaches, of which the details are explained in the following subsections, are proposed in this research to acquire high-quality solutions for the studied lexicographic bottleneck mixed-model assembly line balancing problem.

3.1. GAMS

The mathematical model proposed in Section 2 is coded in GAMS v22.5 optimisation software and solved with CPLEX Solver. The reason for choosing GAMS is that GAMS is a high-level modelling system designed for modelling and solving complex large-scale linear, nonlinear, and mixed-integer optimization problems. The system is available for use on various computer platforms and allows the user to build large maintainable models that can be adapted to new situations. The integrated development environment provided by GAMS is connected to a group of third-party optimisation solvers, one of which is CPLEX. GAMS is run on a 3.46 GHz Intel Core i7 processor with 4 GB RAM to solve the example problem given in Section 2. The lower bound on the total number of workstations, which is 12, is given as input data to the model. The cycle time is set to 10 time units ($CT = 10$) and the proportional demands of models A, B and C are calculated as $d_A = 0.33$, $d_B = 0.50$, and $d_C = 0.17$). The optimum result with 12 workstations is taken after 66 CPU seconds and reported in Table 4.

Table 4. Detailed assignment results of GAMS

Station No	1	2	3	4	5	6	7	8	9	10	11	12
Assigned Tasks	1	3	4	5, 6	7, 14	8	9, 13	10, 11	12, 15	2, 18	16, 19	17, 20, 21
S_k	5.900	7.000	5.217	8.450	8.083	6.653	8.350	5.760	8.170	8.850	8.500	7.919
Workload for Model A	5.90	7.00	5.40	9.70	7.70	6.30	5.00	4.50	7.50	8.85	9.70	7.70
Workload for Model B	5.90	7.00	5.00	7.20	8.50	6.70	10.00	5.15	8.50	8.85	7.30	7.90
Workload for Model C	5.90	7.00	5.50	9.70	7.60	7.20	10.00	10.00	8.50	8.85	9.70	8.40

3.2. Artificial bee colony algorithm

The artificial bee colony algorithm (Karaboga 2005) is a nature-inspired swarm optimisation technique which mimics the foraging behaviour of bees in nature. At the beginning, scout bees randomly search food sources and perform a special dance in front of the hive to attract more bees. Follower bees look for the food source based on the information that is given by this dance. When a food source is exhausted, it is abandoned and new sources are searched by the scout bees. The food requirement of the hive is met by this cycle.

Karaboga and Basturk (2008) compared the performance of artificial bee colony algorithm with that of differential evolution, particle swarm optimization and evolutionary algorithms. According to the simulation experiments performed by Karaboga and Basturk (2008), it was concluded that the artificial bee colony algorithm outperformed these algorithms. As artificial bee colony algorithm was a promising algorithm especially for high-dimensional sophisticated problems, it was selected for solving the LB-MALBP.

Although artificial bee colony algorithm has been applied to a wide range of engineering optimisation and design problems successfully, the number of applications in the entire assembly line balancing field is very limited. Özbakir and Tapkan (2011) and Tapkan *et al*. (2012a, 2012b) used bee colony intelligence and bees algorithms, respectively, to solve zone constrained two-sided assembly line balancing problem, which is a different problem than the one studied in this research. Also, Akpinar and Baykasoğlu (2014) applied bee colony algorithm for solving the mixed-model assembly line balancing problem with traditional objective functions. The proposed artificial bee colony algorithm is explained below using the example given in Section 2.

The algorithm starts with initialisation of the parameters; namely the number of scout bees (S), the number of follower bees (F), the maximum number of iterations ($Maxiter$), life time (LF), and hierarchy parameter (β). The parameters are considered as $S = 35$, $F = 5$, $Maxiter = 50$, $LF = 10$ and $\beta = 100$. Feasible initial solutions, each of which is represented by a scout bee, are built randomly. For example, three initial solutions for the considered numerical example are given in Table 5 where S_k rows denote the weighted workloads of workstations.

The followers of a scout bee are generated through a neighbourhood search mechanism performed by randomly changing the location of a randomly selected task on the scout. For example, the five followers of the Scout Bee-1 given in Table 5 can be generated as in Table 6. In this example, task 21 is moved from 5-*th* position to 9-*th* for the first neighbourhood solution, namely Follower Bee-1. Similarly, task 14 is moved from 9-*th* position to 12-*th* in the second follower bee. Please see the underlined tasks in the table.

Table 5. The initial solutions (scout bees) obtained for the numerical example

	Station No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Scout Bee-1	Assigned Tasks	1	2	3	4,21	5,6	7,14	8	9	11	12,13	10	15	18	16,19	17,20
	S_k	5.900	6.050	7.000	5.887	8.450	8.083	6.653	5.000	4.060	4.020	1.700	7.500	2.800	8.500	7.249
Scout Bee-2	Assigned Tasks	1	2	3	4,21	5,6	7,14	8	9,12	10	13	11	15,16	17,18	20	19
	S_k	5.900	6.050	7.000	5.887	8.450	8.083	6.653	5.670	1.700	3.350	4.060	8.700	4.849	5.200	7.300
Scout Bee-3	Assigned Tasks	1	2	3	4	5,21	7,14	6	8	9,12	10,11	13	15	16,17,18	20	19
	S_k	5.9	6.05	7	5.217	7.520	8.083	1.6	6.653	5.670	5.760	3.35	7.5	6.049	5.2	7.3

Table 6. The followers of Scout Bee-1

Station Number		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Scout Bee-1	Assigned Tasks	1	2	3	4,21	5,6	7,14	8	9	11	12,13	10	15	18	16,19	17,20
	S_k	5.900	6.050	7.000	5.887	8.450	8.083	6.653	5.000	4.060	4.020	1.700	7.500	2.800	8.500	7.249
Follower Bee-1	Assigned Tasks	1	2	3	4	5,6	7,14,21	8	9	11	12,13	10	15	18	16,19	17,20
	S_k	5.900	6.050	7.000	5.217	8.450	8.753	6.653	5.000	4.060	4.020	1.700	7.500	2.800	8.500	7.249
Follower Bee-2	Assigned Tasks	1	2	3	4,21	5,6	7	8	9	11,14	12,13	10	15	18	16,19	17,20
	S_k	5.900	6.050	7.000	5.887	8.450	5.434	6.653	5.000	6.709	4.020	1.700	7.500	2.800	8.500	7.249
Follower Bee-3	Assigned Tasks	1	2	3	4,21	5,6	7	8,14	9	11	12,13	10	15	18	16,19	17,20
	S_k	5.900	6.050	7.000	5.887	8.450	5.434	9.302	5.000	4.060	4.020	1.700	7.500	2.800	8.500	7.249
Follower Bee-4	Assigned Tasks	1	2	3	4	5,6	7,14	8,21	9	11	12,13	10	15	18	16,19	17,20
	S_k	5.900	6.050	7.000	5.217	8.450	8.083	7.323	5.000	4.060	4.020	1.700	7.500	2.800	8.500	7.249
Follower Bee-5	Assigned Tasks	1	2	3	4,21	5,6	7,14	8	9	11	12,13	10	15	18,16,17	20	19
	S_k	5.900	6.050	7.000	5.887	8.450	8.083	6.653	5.000	4.060	4.020	1.700	7.500	6.049	5.200	7.300

* Please note that underlined tasks represent movements.

The performance measure of each scout bee is compared to all of its followers one-by-one using Equation (12), and the scout bees are replaced by their followers if the follower has a better performance measure. The main aim is to minimise δ_{diff} and this is equivalent to the objective function used in the mathematical model given in Section 2. By this way, it is endeavoured to hierarchically minimise the weighted workload of the most heavily loaded workstation, followed by the second most heavily loaded workstation, and so on.

$$\delta_{diff} = \frac{\sum_{k=1}^{K}\left(\beta^{K-k+1} \cdot \sum_{m=1}^{P} d_m \cdot \Delta_{km}\right)}{CT_{best} \cdot \beta^{K-1}}, \tag{12}$$

where Δ_{km} is the positive, null, or negative workload difference in terms of model m in the k-th most heavily loaded workstation between the two solutions which are compared to each other; CT_{best} is the best cycle time of the two solutions; β is a parameter whose value must guarantee the hierarchy of the objectives ($\beta > \max(\Delta_{km} - \Delta_{(k+1)m})$; and d_m is the proportional demand of model m, ($d_m = D_m / \sum_{m=1}^{M} D_m$). Please refer to Pastor (2011) for more details on comparing two balancing solutions and the β parameter used in here.

In our example, the Scout Bee-1 will be replaced with Follower Bee-5 in the next iteration. The performance of all scouts and their followers are evaluated, and the best one among all obtained solutions are kept as the best solution. This is repeated until the maximum number of iterations is reached and the best solution is taken. The best solution obtained for this example problem after 50 iterations is presented in Table 7. As can be seen from the table, proposed algorithm finds the same solution found by GAMS, which is referred to as optimal.

Table 7. The best solution obtained for the numerical example

Station No	1	2	3	4	5	6	7	8	9	10	11	12
Assigned Tasks	1	3	4	5,6	7,14	8	9,13	11,10	12,15	18,2	16,19	17,20,21
S_k	5.9	7	5.217	8.45	8.083	6.653	8.35	5.76	8.17	8.85	8.5	7.919

4. Experimental study

In this section, 20 test problems derived from the literature are solved using CPLEX Solver provided in GAMS v22.5 optimisation software and the artificial bee colony algorithm coded in C# environment. The input data used for the experimental tests are given in Section 4.1. The experiments are conducted on the same computer that was used for numerical examples in Section 3 and the results are reported in Section 4.2.

4.1. Test data

The test problems are taken from Simaria and Vilarinho (2002) and presented in Table 8 with their main characteristics.

Table 8. Data for computational tests

#	Test Problem	N	m	D_m	t_{min}	t_{max}	t_{avg}
1	Bowman	8	A	20	1.8	7.8	3.73
			B	28	1.8	7.9	3.57
2	Bowman	8	A	16	0	10	3.54
			B	24	0	7	4.40
			C	8	0	10	5.87
3	Gökçen and Erel (1998)	11	A	20	1.9	8.8	4.97
			B	28	0	8.7	3.37
4	Gökçen and Erel (1998)	11	A	16	0	9.6	3.56
			B	24	1	9.6	4.59
			C	8	1	9.6	4.59
5	Mitchel	21	A	20	0	9.6	5.29
			B	28	0	9.6	4.46
6	Mitchel	21	A	16	0	7.5	4.06
			B	24	0	7.5	4.19
			C	8	0	10	4.68
7	Simaria and Vilarinho	25	A	20	0	9.6	4.74
			B	28	0	9.4	4.35
8	Simaria and Vilarinho	25	A	16	0	9.9	4.37
			B	24	1	10	4.85
			C	8	1	10	4.87
9	Heskiaoff	28	A	20	0	10	5.46
			B	28	0	10	5.75
10	Heskiaoff	28	A	16	0	10	5.39
			B	24	0	10	5.53
			C	8	0	10	5.78
11	Sawyer	30	A	20	0	9.9	4.49
			B	28	0	9.9	4.46
12	Sawyer	30	A	16	0	9.9	4.65
			B	24	0	9.9	4.40
			C	8	0	9.9	4.79
13	Lutz1	32	A	20	0	9.5	3.85
			B	28	0	10	4.21
14	Lutz1	32	A	16	0	9.7	4.60
			B	24	0	9.5	4.40
			C	8	0	9.7	4.61
15	Gunther	35	A	20	0	8.2	4.97
			B	28	0	9	4.88
16	Gunther	35	A	16	0	8.7	4.66
			B	24	0	8.7	4.84
			C	8	0	8.8	4.90
17	Kilbridge and Wester	45	A	20	0	10	4.63
			B	28	0	10	4.53
18	Kilbridge and Wester	45	A	16	0	9.3	4.64
			B	24	0	9.3	4.61
			C	8	0	9.3	4.24
19	Tonge	70	A	20	0	9.9	4.85
			B	28	0	10	4.99
20	Tonge	70	A	16	0	9.7	5.02
			B	24	0	9.7	5.02
			C	8	0	9.6	5.03

The second and third columns correspond to the name of the test problem and the number of tasks that it has in its combined precedence relationships diagram, respectively. D_m column gives the demand of model given in column m. The columns t_{min}, t_{max} and t_{avg} present the minimum, maximum and average processing times of the tasks for the models of the considered test problems, respectively. CT is assumed 10 for all test problems ($CT = 10$). If a task's processing time belonging to any of the models exceeds the cycle time, processing times of that task are divided into two ($t_{im} \leftarrow t_{im}/2$) for all models. This is because the parallel workstations are not considered as different from the study of Simaria and Vilarinho (2002).

4.2. Test results

The test problems, for which the details are given in Section 4.1, are solved through two ways: *(i)* the proposed mathematical model coded in GAMS v22.5, and *(ii)* the proposed artificial bee colony algorithm coded in C#. Table 9 reports the results of the experimental tests obtained from GAMS. As could be seen from the table, the results are reported only for the first six test problems as the software did not produce any solution for the rest of the problems in 48 hours due to increasing complexity when the number of tasks increases. These solutions are referred to as optimal as they are investigated by the mathematical model proposed in this work.

Table 9. Optimal results obtained by the mathematical model coded in GAMS

#	K	CPU (min.)	δ	Weighted Workloads of the Workstations (in descending order)
1	4	0.001	79.34	7.858, 7.500, 7.117, 6.600
2	8	0.015	78.87	7.825, 6.150, 4.690, 4.020, 3.900, 3.652, 3.000, 1.700
3	7	0.028	87.85	8.700, 8.400, 6.584, 6.260, 5.600, 5.223, 3.696
4	5	0.008	81.86	10.134, 9.790, 9.600, 8.917, 8.330
5	13	1469	97.98	9.700, 9.700, 9.600, 8.936, 8.500, 8.038, 7.738, 7.700, 7.482, 6.474, 6.400, 5.800, 4.998
6	12	1.060	89.36	8.850, 8.500, 8.450, 8.350, 8.170, 8.083, 7.919, 7.000, 6.653, 5.900, 5.760, 5.217

The *K* and *CPU* columns give the number of workstations obtained and the CPU time needed to solve the relevant test problem using the proposed method, respectively. As there is no published work which addresses the lexicographic bottleneck mixed-model assembly line balancing problem in the literature, there is no available result to compare our results. For that reason, we also calculate the performance measure of each single solution using the Equation (13) and present in δ column. Thus, we provide results which can be comparable in future studies. Weighted workloads of the workstations are also given in descending order for the optimal results.

$$\delta = \frac{\sum_{k=1}^{K}\left(\beta^{K-k+1} \cdot S_k\right)}{CT \cdot \beta^{K-1}}. \qquad (13)$$

The parameters of the artificial bee colony algorithm are determined through a set of comprehensive experimental tests to achieve high-quality solutions as the complexity of the problem increases in conjunction with the increasing number of tasks. Each test problem is solved using various levels of parameters, presented in Table 10, and the best solution among the obtained results is reported in Table 11 (where $\beta = 100$ for all test problems). Therefore, the parameter set which gives the best solution is also reported for each problem, as different from the GAMS results. If the same balancing solution is obtained from two or more different sets of parameters, the solution which requires the minimum CPU time is reported.

Table 10. The various levels of parameters used for the artificial bee colony algorithm

S	F	Maxiter	LF
5	5	50	10
20	15	175	25
35	25	300	40

While the *Maxiter* column denotes the maximum number of iterations that the artificial bee colony algorithm is run, the *IT* column represents the iteration number in which the best solution is observed. Weighted workloads of the workstations are also reported in descending order for each test problem.

When the solutions obtained by GAMS (in Table 9) and artificial bee colony algorithm (in Table 11) are compared to each other, it is clear that the proposed artificial bee colony algorithm performs quite well. The proposed approach investigates optimal results for five test problems (i.e. 1, 2, 3, 5, 6) out of six, solved by GAMS. For test problem 4, the proposed approach finds a solution with performance measures of $\delta = 96.85$ and $K = 6$ while $\delta = 81.86$ and $K = 5$ for the optimal result. For the remaining test problems (7-20), the proposed approach finds solutions with reasonably well δ values, mostly lower than 100. Higher δ values are observed for only four test problems, *i.e.* 9, 10, 11, 15.

Table 11. Test results obtained from the proposed artificial bee colony algorithm

#	Parameters				K	IT	CPU (sec.)	δ	Weighted Workloads of the Workstations (in descending order)
	S	F	Maxiter	LF					
1	5	5	50	40	4	1	0.04	79.34	7.858, 7.5, 7.259, 6.458
2	5	5	50	10	8	1	0.05	78.87	7.825, 6.15, 4.69, 4.02, 3.9, 3.652, 3, 1.7
3	5	5	50	40	7	1	0.03	87.85	8.7, 8.4, 6.584, 6.26, 5.6, 5.223, 3.696
4	5	5	50	10	6	1	0.02	96.85	9.6, 8.45, 7.84, 7.8, 7.051, 6.03
5	5	5	50	10	13	36	0.06	97.98	9.7, 9.7, 9.6, 8.936, 8.5, 8.038, 7.7, 7.482, 7.438, 6.474, 6.4, 6.1, 4.998
6	35	5	50	10	12	33	0.42	89.36	8.85, 8.5, 8.45, 8.35, 8.17, 8.083, 7.919, 7, 6.653, 5.9, 5.76, 5.217
7	5	25	50	10	15	34	0.31	94.89	9.4, 8.8, 8.788, 8.568, 8.46, 8.4, 7.858, 7.7, 7.5, 7.5, 7.146, 7.012, 6.774, 6.264, 2.646
8	35	5	50	10	14	38	0.45	99.94	9.9, 9.349, 9.2, 9.15, 9.14, 8.8, 8.7, 8.687, 8.617, 7.8, 7.8, 7.2, 6.7, 6.299
9	35	5	50	10	19	13	0.37	100.99	10, 9.8, 9.732, 9.7, 9.6, 9.6, 9.5, 9.2, 9.162, 9.058, 8.9, 8.7, 8.2, 8.1, 7.6, 6.348, 5.22, 5.22, 4.158
10	5	5	50	10	19	22	0.07	101.00	10, 9.9, 9.585, 9.3, 9.25, 9.2, 9.048, 8.972, 8.6, 8.6, 8.6, 8.4, 8, 7.968, 7.7, 6.717, 6.03, 4.933, 4.1
11	20	15	175	25	15	175	2.20	101.00	10, 9.9, 9.742, 9.7, 9.5, 9.4, 9.2, 9.07, 8.96, 8.7, 8.616, 8.6, 8.5, 8.284, 6.048
12	20	15	175	25	17	14	1.19	99.98	9.9, 9.7, 9.4, 9.251, 9.15, 9.1, 7.825, 7.8, 7.75, 7.734, 7.7, 7.64, 7.518, 7.2, 6.9, 6.806, 5.24
13	5	5	50	10	17	32	0.04	95.94	9.5, 9.3, 9.142, 9.068, 8.574, 8.248, 8.242, 8.016, 8.006, 8, 6.9, 6.626, 6.6, 6.55, 6.1, 5.8, 5.298
14	5	25	50	10	18	12	0.24	97.46	9.65, 9.5, 9.3, 9.2, 9.15, 8.95, 8.6, 8.4, 8.25, 8.15, 8.083, 7.76, 7.742, 7.7, 7.5, 6.832, 5.65, 3.75
15	20	15	175	25	21	109	1.43	101.01	10, 10, 9.8, 9.4, 9.316, 9.3, 8.8, 8.7, 8.3, 8.24, 8.2, 8.2, 8.152, 7.544, 7.4, 7.362, 7.2, 7.064, 6.9, 6.8, 5.624
16	35	25	300	10	20	258	8.30	99.95	9.9, 9.4, 9.251, 9, 8.867, 8.7, 8.7, 8.619, 8.58, 8.532, 8.385, 8.316, 8.3, 8.2, 8.05, 8, 7.902, 7.4, 7.1, 6.46
17	35	25	300	10	24	169	18.08	98.97	9.8, 9.6, 9.506, 9.4, 9.4, 9.362, 9.306, 9.206, 9.178, 9.048, 9.042, 9, 8.824, 8.8, 8.6, 8.54, 8.384, 8.3, 8.2, 8.134, 8, 7.46, 7.308, 3.36
18	20	15	50	25	26	24	0.54	93.93	9.3, 9.2, 9, 8.8, 8.7, 8.6, 8.25, 8.2, 8.122, 8.1, 7.978, 7.9, 7.878, 7.8, 7.75, 7.719, 7.667, 7.6, 7.583, 7.5, 7.47, 7.398, 7, 6.56, 6.491, 6.374
19	20	15	175	25	43	105	3.42	99.97	9.9, 9.6, 9.6, 9.4, 9.3, 9.2, 9, 8.9, 8.9, 8.81, 8.8, 8.8, 8.782, 8.736, 8.7, 8.5, 8.5, 8.4, 8.4, 8.3, 8.2, 8.2, 8.166, 8.1, 8, 7.958, 7.916, 7.9, 7.7, 7.58, 7.5, 7.5, 7.396, 7.3, 7.064, 6.8, 6.732, 6.606, 6.526, 6.202, 5.926, 5.8, 5.7
20	35	15	175	25	43	152	10.42	96.96	9.6, 9.5, 9.5, 9.231, 9.2, 8.997, 8.901, 8.9, 8.9, 8.9, 8.885, 8.85, 8.8, 8.733, 8.7, 8.7, 8.516, 8.4, 8.4, 8.4, 8.4, 8.383, 8.2, 8.15, 8.1, 8.1, 8.1, 8.058, 8.051, 8, 7.92, 7.9, 7.9, 7.65, 7.65, 7.6, 7.554, 7.4, 7.257, 7.15, 6.7, 5.3, 3.85

It can be seen that the iteration number that the best solution is found (*IT*) increases based on the increasing complexity of the test problems when the number of tasks (*N*) increases. However, the CPU times consumed by the algorithm are quite reasonable even for the large-sized problems. The largest and the second largest CPU times are observed 18.08 and 10.42 seconds for test problems 17 and 20, respectively. Therefore, it can be argued that the proposed artificial bee colony algorithm produces good quality solutions in quite reasonable CPU times.

5. Conclusions and future research

The main contribution of this study is that the lexicographic bottleneck mixed-model assembly line balancing problem is defined and mathematically formulated for the first time in the literature. Numerical examples are provided with alternative balancing solutions for the detailed description of the problem. The other contributions of the current work are the proposed solution methods. The proposed mathematical model is coded in GAMS v22.5 to get optimal solutions. Also, a new powerful artificial bee colony algorithm is developed and coded in C# to efficiently solve the problems, which cannot be solved by GAMS in today's computer technology, by dealing with the variations in processing times of tasks among different product models. The solution building mechanism of the proposed algorithm is described step-by-step through a numerical example. An experimental study is carried out solving twenty test problems derived from the literature. When the results obtained by GAMS and the artificial bee colony algorithm are compared to each other, it is observed that the proposed approach finds optimal solutions for majority of the test problems solved by GAMS. In accordance with the performance measures of the obtained solutions, it can be argued that the proposed artificial bee colony algorithm provides quite promising results by consuming quite reasonable CPU times, especially for the large-sized test problems.

As a possible extension and practical implication of this study, the proposed model in this work can be applied in real world applications and the physical outcomes (advantages and disadvantages) of the system can be observed. The line managers of actual production line systems can take advantage of the proposed line balancing methodology. Also, the proposed artificial bee colony algorithm can be enhanced with several local search procedures and its performance can be compared to some other well-known meta-heuristics such as tabu search algorithm, ant colony optimisation algorithm and genetic algorithm; and recently developed algorithms such as gravitational search algorithm (Rashedi, Nezamabadi-pour, and Saryazdi 2009), Jaya (Venkata Rao 2016) and great deluge algorithm (Dueck 1993).

References

Akgunduz, O. S., and S. Tunali. 2010. "An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem." *International Journal of Production Research* 48 (17):5157-5179.

Akpinar, S., G. M. Bayhan, and A. Baykasoglu. 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.

Akpinar, S., and A. Baykasoglu. 2014. "Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model." *Journal of Manufacturing Systems* 33 (1):177-187.

Askin, R. G., and M. Zhou. 1997. "A parallel station heuristic for the mixed-model production line balancing problem." *International Journal of Production Research* 35 (11):3095-3106.

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

Battini, D., M. Faccio, E. Ferrari, A. Persona, and F. Sgarbossa. 2007. "Design configuration for a mixed-model assembly system in case of low product demand." *International Journal of Advanced Manufacturing Technology* 34 (1-2):188-200.

Baybars, I. 1986. "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem." *Management Science* 32 (8):909-932.

Baykasoglu, A., and T. Dereli. 2009. "Simple and U-Type Assembly Line Balancing by Using an Ant Colony Based Algorithm." *Mathematical & Computational Applications* 14 (1):1-12.

Becker, C., and A. Scholl. 2006. "A survey on problems and methods in generalized assembly line balancing." *European Journal of Operational Research* 168 (3):694-715.

Boysen, N., M. Fliedner, and A. Scholl. 2007. "A classification of assembly line balancing problems." *European Journal of Operational Research* 183 (2):674-693.

Chutima, P., and P. Chimklai. 2012. "Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge." *Computers & Industrial Engineering* 62 (1):39-55.

Dueck, Gunter. 1993. "New Optimization Heuristics." *Journal of Computational Physics* 104 (1):86-92.

Emde, S., N. Boysen, and A. Scholl. 2010. "Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload." *International Journal of Production Research* 48 (11):3173-3191.

García-Villoria, Alberto, and Rafael Pastor. 2013. "Erratum to “A solution procedure for type E simple assembly line balancing problem”." *Computers & Industrial Engineering* 66 (1):201-202.

Ghosh, S., and R. J. Gagnon. 1989. "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems." *International Journal of Production Research* 27 (4):637-670.

Gokcen, H., and E. Erel. 1997. "A goal programming approach to mixed-model assembly line balancing problem." *International Journal of Production Economics* 48 (2):177-185.

Gökçen, H., and E. Erel. 1998. "Binary integer formulation for mixed-model assembly line balancing problem." *Computers & Industrial Engineering* 34 (2):451-461.

Hackman, Steven T., Michael J. Magazine, and T. S. Wee. 1989. "Fast, Effective Algorithms for Simple Assembly Line Balancing Problems." *Operations Research* 37 (6):916-924.

Hamta, N., S. M. T. F. Ghomi, F. Jolai, and M. A. Shirazi. 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.

Hamzadayi, A., and G. Yildiz. 2012. "A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints." *Computers & Industrial Engineering* 62 (1):206-215.

Haq, A. N., K. Rengarajan, and J. Jayaprakash. 2006. "A hybrid genetic algorithm approach to mixed-model assembly line balancing." *International Journal of Advanced Manufacturing Technology* 28 (3-4):337-341.

Hwang, R., and H. Katayama. 2009. "A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems." *International Journal of Production Research* 47 (14):3797-3822.

Hwang, R., and H. Katayama. 2010. "Integrated procedure of balancing and sequencing for mixed-model assembly lines: a multi-objective evolutionary approach." *International Journal of Production Research* 48 (21):6417-6441.

Kara, Y., U. Ozcan, and A. Peker. 2007a. "An approach for balancing and sequencing mixed-model JIT U-lines." *International Journal of Advanced Manufacturing Technology* 32 (11-12):1218-1231.

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

Kara, Y., and M. Tekin. 2009. "A mixed integer linear programming formulation for optimal balancing of mixed-model U-lines." *International Journal of ProductionResearch* 47 (15):4201-4233.

Karaboga, D. 2005. "An idea based on honey bee swarm for numerical optimization. Technical Report TR06." In. Computer Engineering Department, Engineering Faculty, Erciyes University, Turkey.

Karaboga, D., and B. Basturk. 2008. "On the performance of artificial bee colony (ABC) algorithm." *Applied Soft Computing* 8 (1):687-697.

Kucukkoc, I., A. D. Karaoglan, and R. Yaman. 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, doi: http://dx.doi.org/10.1080/00207543.2013.784411.

Kucukkoc, I., and D. Z. Zhang. 2014a. "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, doi: http://dx.doi.org/10.1016/j.ijpe.2014.08.010.

Kucukkoc, I., and D. Z. Zhang. 2014b. "Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines." *International Journal of Production Research* 52 (12):3665-3687, doi: http://dx.doi.org/10.1080/00207543.2013.879618.

Kucukkoc, I., and D. Z. Zhang. 2015a. "Balancing of Parallel U-shaped Assembly Lines." *Computers and Operations Research* 64:233–244, doi: http://dx.doi.org/10.1016/j.cor.2015.05.014.

Kucukkoc, I., and D. Z. Zhang. 2015b. "A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem." *Production Planning and Control* 26 (11):874–894, doi: http://dx.doi.org/10.1080/09537287.2014.994685.

Kucukkoc, I., and D. Z. Zhang. 2015c. "Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters." *Computers and Industrial Engineering* 84:56–69, doi: http://dx.doi.org/10.1016/j.cie.2014.12.037.

Kucukkoc, Ibrahim, and DavidZ Zhang. 2015d. "Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines." *The International Journal of Advanced Manufacturing Technology*:doi: http://dx.doi.org/10.1007/s00170-015-7320-y.

Liao, L. M., C. J. Huang, and J. H. Huang. 2012. "Applying Multi-agent Approach to Mixed-model Assembly Line Balancing." In *Proceedings of the IEEE ICMIT*.

Liu, S. B., H. L. Ong, and H. C. Huang. 2004. "A bidirectional heuristic for stochastic assembly line balancing Type II problem." *The International Journal of Advanced Manufacturing Technology* 25 (1-2):71-77.

Manavizadeh, N., N. S. Hosseini, M. Rabbani, and F. Jolai. 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.

Manavizadeh, N., M. Rabbani, D. Moshtaghi, and F. Jolai. 2012. "Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms." *Expert Systems with Applications* 39 (15):12026-12031.

McMullen, P. R., and P. Tarasewich. 2003. "Using ant techniques to solve the assembly line balancing problem." *IIE Transactions* 35 (7):605-617.

Mosadegh, H., M. Zandieh, and S. M. T. F. Ghomi. 2012. "Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines." *Applied Soft Computing* 12 (4):1359-1370.

Ozbakir, L., and P. Tapkan. 2011. "Bee colony intelligence in zone constrained two-sided assembly line balancing problem." *Expert Systems with Applications* 38 (9):11947-11957.

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

Ozcan, U., and B. Toklu. 2009. "Balancing of mixed-model two-sided assembly lines." *Computers & Industrial Engineering* 57 (1):217-227.

Pastor, Rafael. 2011. "LB-ALBP: the lexicographic bottleneck assembly line balancing problem." *International Journal of Production Research* 49 (8):2425-2442.

Pastor, Rafael, Ignacio Chueca, and Alberto García-Villoria. 2012. "A heuristic procedure for solving the Lexicographic Bottleneck Assembly Line Balancing Problem (LB-ALBP)." *International Journal of Production Research* 50 (7):1862-1876.

Pastor, Rafael, Alberto García-Villoria, Manuel Laguna, and Rafael Martí. 2015. "Metaheuristic procedures for the lexicographic bottleneck assembly line balancing problem." *Journal of the Operational Research Society* doi: http://dx.doi.org/10.1057/jors.2014.138.

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

Rashedi, Esmat, Hossein Nezamabadi-pour, and Saeid Saryazdi. 2009. "GSA: A Gravitational Search Algorithm." *Information Sciences* 179 (13):2232-2248.

Rekiek, B., A. Dolgui, A. Delchambre, and A. Bratcu. 2002. "State of art of optimization methods for assembly line design." *Annual Reviews in Control* 26:163-174.

Satoglu, Sule Itir, and I. Ethem Sahin. 2012. "Design of a just-in-time periodic material supply system for the assembly lines and an application in electronics industry." *The International Journal of Advanced Manufacturing Technology* 65 (1-4):319-332.

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

Simaria, Ana Sofia, and Pedro M. Vilarinho. 2004. "A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II." *Computers & Industrial Engineering* 47 (4):391-407.

Sivasankaran, P., and P. Shahabudeen. 2014. "Literature review of assembly line balancing problems." *The International Journal of Advanced Manufacturing Technology* 73 (9-12):1665-1694.

Tapkan, P., L. Ozbakir, and A. Baykasoglu. 2012a. "Bees Algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem." *Optimization Letters* 6 (6):1039-1049.

Tapkan, P., L. Ozbakir, and A. Baykasoglu. 2012b. "Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms." *Applied Soft Computing* 12 (11):3343-3355.

Thomopoulos, N. T. 1967. "Line Balancing-Sequencing for Mixed-Model Assembly." *Management Science* 14 (2):59-75.

Venkata Rao, R. 2016. "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems." *International Journal of Industrial Engineering Computations* 7:19-34, doi: http://dx.doi.org/10.5267/j.ijiec.2015.8.004.

Vilarinho, P. M., and A. S. Simaria. 2002. "A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations." *International Journal of Production Research* 40 (6):1405-1420.

Wei, Nai-Chieh, and I. Ming Chao. 2011. "A solution procedure for type E simple assembly line balancing problem." *Computers & Industrial Engineering* 61 (3):824-830.

Xu, W. , and T. Xiao. 2011. "Strategic Robust Mixed Model Assembly Line Balancing Based on Scenario Planning." *Tsinghua Science and Technology* 16 (3):308-314.

Yagmahan, B. 2011. "Mixed-model assembly line balancing using a multi-objective ant colony optimization approach." *Expert Systems with Applications* 38 (10):12453-12461.

Yoosefelahi, A., M. Aminnayeri, H. Mosadegh, and H. Davari Ardakani. 2012. "Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model." *Journal of Manufacturing Systems* 31 (2):139-151.

Zhang, D. Z., and I. Kucukkoc. 2013. "Balancing Mixed-Model Parallel Two-Sided Assembly Lines." In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management, IEEE - IESM 2013, art. no. 6761429, 28-30 October, Rabat, Morocco*, edited by L. Amodeo, A. Dolgui and F. Yalaoui, 391-401.

Zhang, W. Q., and M. Gen. 2011. "An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time." *Journal of Intelligent Manufacturing* 22 (3):367-378.