Taylor & Francis
Taylor & Francis Group

Production Planning & Control
The Management of Operations

ISSN: 0953-7287 (Print) 1366-5871 (Online) Journal homepage: https://www.tandfonline.com/loi/tppc20

A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem

Ibrahim Kucukkoc & David Z. Zhang

To cite this article: Ibrahim Kucukkoc & David Z. Zhang (2015) A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem, Production Planning & Control, 26:11, 874-894, DOI: 10.1080/09537287.2014.994685

To link to this article: https://doi.org/10.1080/09537287.2014.994685

Published online: 27 Apr 2015.

Submit your article to this journal

Article views: 2416

View related articles

View Crossmark data

Citing articles: 17 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tppc20

Production Planning & Control, 2015
Vol. 26, No. 11, 874–894, http://dx.doi.org/10.1080/09537287.2014.994685

Taylor & Francis
Taylor & Francis Group

A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem

Ibrahim Kucukkoc[a,b]* and David Z. Zhang[a]

[a]*College of Engineering, Mathematics and Physical Sciences, University of Exeter, Streatham Campus, North Park Road, EX4 4QF Exeter, England, UK;* [b]*Faculty of Engineering and Architecture, Department of Industrial Engineering, Balikesir University, Cagis Campus, 10145 Balikesir, Turkey*

(*Received 3 July 2014; accepted 8 November 2014*)

Assembly lines are usually constructed as the last stage of the entire production system and efficiency of an assembly line is one of the most important factors which affect the performance of a complex production system. The main purpose of this paper is to mathematically formulate and to provide an insight for modelling the parallel two-sided assembly line balancing problem, where two or more two-sided assembly lines are constructed in parallel to each other. We also propose a new genetic algorithm (GA)-based approach in alternatively to the existing only solution approach in the literature, which is a tabu search algorithm. To the best of our knowledge, this is the first formal presentation of the problem as well as the proposed algorithm is the first attempt to solve the problem with a GA-based approach in the literature. The proposed approach is illustrated with an example to explain the procedures of the algorithm. Test problems are solved and promising results are obtained. Statistical tests are designed to analyse the advantage of line parallelisation in two-sided assembly lines through obtained test results. The response of the overall system to the changes in the cycle times of the parallel lines is also analysed through test problems for the first time in the literature.

Keywords: parallel two-sided assembly lines; assembly line balancing; production planning; genetic algorithm; meta-heuristics; artificial intelligence

1. Introduction

Assembly lines have been utilised successfully to produce large-volume high-quality standardised homogeneous products, and have been of interest for both academia and industry for decades. An assembly line is a sequential organisation of workstations (or operators) linked by a conveyor belt or material handling system on which semi-finished products are moved from one workstation to another (Ozdemir and Ayag 2011). Parts are added on the moving semi-finished products in sequence until the final assembly is produced. A group of tasks is performed in each workstation by considering capacity of the workstation and precedence relationships among tasks, where precedence relationships are usually caused by the technological requirements or organisational structures (Tonelli et al. 2013; Kucukkoc and Zhang 2014a). Sum of processing times of all tasks assigned to a workstation constitutes its workload time and workload time of a workstation cannot exceed the designated cycle time. Assembly line balancing problem is to determine which task will be accomplished in which workstation by assigning tasks to an ordered sequence of workstations considering aforementioned constraints (i.e. capacity constraints, assignment constraints, precedence relationships constraints, etc.). Tasks cannot be split into two or more pieces and each task must be assigned to exactly one workstation (Kucukkoc, Karaoglan, and Yaman 2013; Buyukozkan, Kucukkoc, and Zhang 2014).

In accordance with the utilisation of operation sides, assembly lines can be classified as one-sided assembly lines and two-sided assembly lines. Tasks are performed on both left and right sides of the line in a two-sided assembly line system, while only left or right side of the line is used in a one-sided assembly line system. Two-sided assembly lines are usually utilised to produce high-volume large-sized products, such as trucks and buses (Kucukkoc and Zhang 2014c). Some heuristic approaches were proposed by Lee, Kim, and Kim (2001), Hu, Wu, and Jin (2008), Ozcan and Toklu (2010) and Yegul, Agpak, and Yavuz (2010); and some exact solution approaches were developed by Wu et al. (2008) and Hu et al. (2010), since the two-sided assembly line balancing problem was first introduced by Bartholdi (1993). Meta-heuristics have also been presented by Baykasoglu and Dereli (2008), Simaria and Vilarinho (2009), Ozcan and Toklu (2009), Ozbakir and Tapkan (2010), Ozcan (2010), Ozbakir and Tapkan (2011),

*Corresponding author. Emails: i.kucukkoc@exeter.ac.uk, ikucukkoc@balikesir.edu.tr

© 2015 Taylor & Francis

Chutima and Chimklai (2012) and Khorasanian, Hejazi, and Moslehi (2013). Among these meta-heuristics, studies by Kim, Kim, and Kim (2000), Kim, Song, and Kim (2009), Taha et al. (2011), Rabbani, Moghaddam, and Manavizadeh (2012) and Purnomo, Wee, and Rau (2013) employed genetic algorithm (GA)-based approaches to balance two-sided lines. As can be comprehended from these studies, there exist numerous successfully implemented GA-based approaches in the literature for two-sided assembly line balancing problems.

There is another type of line configuration called parallel assembly line system, where two or more lines are located in parallel to each other to maximise the use of shared recourses and tools. The idea of balancing more than one assembly line with a common set of resources was first introduced by Gökçen, Agpak, and Benzer (2006). Gökçen, Agpak, and Benzer (2006) proposed new procedures and a mathematical model on the single model assembly line balancing problem with parallel lines. Few researchers followed Gökçen, Agpak, and Benzer (2006) and a novel ant colony optimisation-based algorithm was proposed by Baykasoglu et al. (2009) for the *parallel assembly line balancing problem (PALBP)*. Cercioglu et al. (2009) proposed a simulated annealing approach to solve PALBP and compared results obtained from the algorithm with the results of existing heuristic algorithm proposed by Gökçen, Agpak, and Benzer (2006). The first multi-objective tabu search algorithm for PALBP was presented by Ozcan et al. (2009) and its performance was tested on a set of well-known problems in the literature. Another mathematical model of the PALBP was developed by Scholl and Boysen (2009) along with an exact solution procedure. Kara, Gokcen, and Atasagun (2010) suggested a fuzzy goal programming model that can be used for balancing parallel assembly lines. Ozcan et al. (2010) addressed parallel mixed-model assembly line balancing and sequencing problem with a simulated annealing approach to maximise the line efficiency by ensuring smooth workload distribution among workstations. Ozbakir et al. (2011) developed a multiple-colony ant algorithm for balancing bi-objective parallel assembly lines, while Kucukkoc and Zhang (2014b, 2014c) considered the model sequencing problem as well as the line balancing problem on mixed-model parallel two-sided assembly lines and proposed agent-based ant colony optimisation solution approaches. Please refer to Lusa (2008) and Zhang and Kucukkoc (2013) for a detailed survey on multiple and PALBPs.

The combination of above-mentioned configurations, namely *parallel two-sided assembly lines*, is also frequently constructed in industry in production of large-sized items. Although vast numbers of researches have been carried out on traditional configurations of assembly lines in the literature, there is only one research concerning *parallel two-sided assembly line balancing problem (PTALBP)*. The concept of parallel two-sided assembly lines was first described by Ozcan, Gokcen, and Toklu (2010). Ozcan, Gokcen, and Toklu (2010) introduced and defined the PTALBP and proposed a tabu search algorithm to solve the combined well-known test problems in the literature. Obtained results were compared with theoretical minimum number of workstations to show the performance of the proposed algorithm. The research also demonstrated that parallelisation of two-sided lines helps to lower the total number of workstations, but no statistical technique was used for this aim.

GAs as well as other evolutionary approaches have been applied to line balancing problems earlier with success. There is continuing work in applying GA for various types of line balancing problems, i.e. Leu, Matheson, and Rees (1994), Rubinovitz and Levitin (1995), Kim, Kim, and Kim (2000), Rekiek et al. (2001), Goncalves and de Almeida (2002), Simaria and Vilarinho (2004), Zhang, Kan, and Wang (2005), Haq, Rengarajan, and Jayaprakash (2006), Levitin, Rubinovitz, and Shnits (2006), Suwannarongsri, Limnararat, and Puangdownreong (2007), Zhang, Gen, and Lin (2008), Hwang and Katayama (2009), Yu and Yin (2010), Chica, Cordon, and Damas (2011), Akpinar and Bayhan (2011) and Kucukkoc, Karaoglan, and Yaman (2013). In particular, Kim, Kim, and Kim (2000), Kim, Song, and Kim (2009), Taha et al. (2011), Rabbani, Moghaddam, and Manavizadeh (2012) and Purnomo, Wee, and Rau (2013) have solved two-sided assembly line balancing problems with different GA-based approaches, but none of them have considered line parallelisation as an additional specification. On the contrary of its successful implementations on various line systems, PTALBP has not been addressed using any GA-based technique. Therefore, there is neither GA based nor evolutionary approach published concerning parallel two-sided assembly lines. This is the main motivation of why a GA-based approach is proposed in solving the addressed problem in this research.

Evidence of the need for this research is shown by the lack of literature on developing the mathematical model of the PTALBP and presenting the positive effect of line parallelisation on two-sided lines, statistically. The need for this research is also guided by the gap in the literature on balancing more than one two-sided assembly line with a common set of resources using an evolutionary-based approach, such as GA. Moreover, the response of the whole line system against the changes in the cycle times of the parallel lines is demonstrated for the first time in the literature. From the managerial point of view, among the available solutions, line managers can easily pick up a solution for a specific combination of cycle times of the parallel lines. This helps them make decision especially when there is change in model demands.

The rest of the paper is organised as follows. Section 2 presents the main characteristics of the problem along with the mathematical model and assumptions considered. The proposed approach is described in Section 3 and illustrated with an example in Section 4. Section 5 reports and statistically analyses the results of the computational study. Section 6 concludes with the findings of the research and the future research directions. Some graphs related to the computational study are also depicted in the Appendix 1.

2. Problem statement

2.1. *Main characteristics*

To maximise the use of shared tools and minimise idle times of the entire production system, two or more two-sided lines – on which two or more similar product models that have similar production processes are produced – are located in parallel to each other. Such a configuration is called parallel two-sided assembly line system and can be typically illustrated as in Figure 1. The PTALBP is balancing two or more two-sided assembly lines, which are constructed in parallel to each other. A common set of resources is shared among the lines and the main objective is allocating tasks to the workstations to optimise a performance measure (i.e. number of workstations, line efficiency, etc.) by considering technological priorities, capacity constraints and some other possible constraints caused by organisational structures or technological requirements. Different product models are produced on each of the two-sided assembly lines, represented by h (where $h = 1, \ldots, H$); and each product model has its own set of tasks ($i = 1, \ldots, n_h$). These tasks are performed according to the known precedence relationships among tasks. P_h represents a set of precedence relationships on line h, where $(r, s) \in P_h$ represents that Task-r must be completed to be able to assign Task-s. Each task, which is performed on line h, needs a certain amount of processing time symbolised with t_{hi}; and each line consists of a series of workstations ($k = 1, \ldots, K$) (Ozcan, Gokcen, and Toklu 2010).

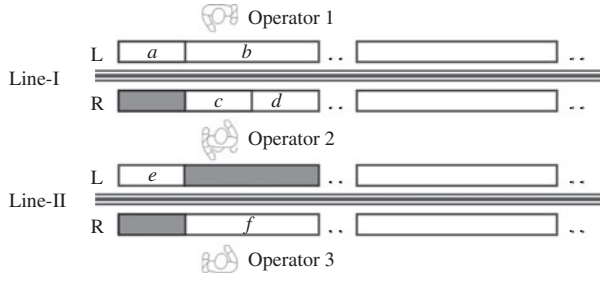


Figure 1. Typical illustration of parallel two-sided assembly lines, adapted from (Ozcan, Gokcen, and Toklu 2010).

The main advantage of parallel two-sided assembly lines is the flexibility of utilising multi-line stations between two adjacent lines. Operators located in stations between two adjacent lines can perform tasks from both lines. By this way, idle times are reduced and system utilisation is increased. As can be seen in Figure 1, three operators are needed to perform Task-*a*–Task-*f*, and Operator-2 first completes Task-*e* at the left-side station of Line-II, and then Task-*c* and Task-*d* at the right-side station of Line-I. Please note that the shades in the figure symbolise idle times (Ozcan, Gokcen, and Toklu 2010).

It should be noted that more attention is needed when balancing two-sided assembly lines, because tasks, which have precedence relationships with each other and are performed on different sides of the lines, must be assigned considering the finishing time of previously assigned tasks. Let P_1 denotes the set of precedence relationships of Line-I. If $(a, b) \in P_1$ and $(a, c) \in P_1$, then Task-*b* and Task-*c* can be initialised after completion of Task-*a*, which may be performed at the other side of the line. This phenomenon is called *interference* in the literature and the violation of this rule yields infeasible balancing solutions.

Another significant advantage of this line system is that each line may have a different cycle time (C_h), which means that each line may have a different throughput rate contributing to the flexibility. When two lines which have different cycle time are subject to balancing, a common cycle time should be used to assign tasks in each cycle. Gökçen, Agpak, and Benzer (2006) used least common multiple (*LCM*)-based approach for different cycle time situations of two parallel lines (Ozcan, Gokcen, and Toklu 2010). In this approach, (Gökçen, Agpak, and Benzer 2006):

- LCM of the cycle times is found.
- Line divisors (ld_1 and ld_2) are calculated through dividing the *LCM* value by the cycle times of Line-I and Line-II (C_1 and C_2), respectively.
- Task times of the product models produced on the Line-I and Line-II are multiplied by ld_1 and ld_2, separately.
- *LCM* is determined as the common cycle time (C) of the lines and the lines are balanced together.

To characterise the PTALBP more clearly and provide an insight for modelling of PTALBP and utilisation of multi-line stations, a numerical example is given below. Data given in Table 1 are used as input for the example problem and a possible balancing solution for the considered problem is exhibited in Figure 2 under 12 time units cycle time constraint for both the lines.

Figure 2 shows the utilisation of multi-line stations between the adjacent two-sided lines located in parallel to each other. Numbers inside bars denote task numbers, while lengths of the bars correspond to process-

Table 1. Data for numerical example.

	Line-I			Line-II		
Task no	Side	Processing time	Immediate predecessors	Side	Processing time	Immediate predecessors
1	Left	2	–	Left	2	–
2	Either	4	1	Right	2	–
3	Right	3	–	Left	4	–
4	Either	3	2	Either	1	2
5	Left	6	1	Right	3	2
6	Either	4	5	Either	3	1
7	Left	5	4,6	Left	5	3,4,6
8	Either	1	4	Either	4	5
9	Either	3	8	Left	4	7
10	Right	2	3	Either	3	7,8
11	Either	2	10	Either	3	–
12	Left	4	11	Left	2	9
13	Either	3	7	Right	5	10
14	Left	4	13	Either	3	11
15	Either	2	12	Left	6	12
16	Either	3	14,15	Either	3	13,15
17	Left	2	16	Either	6	14
18	–	–	–	Left	2	16
Total time		53			61	

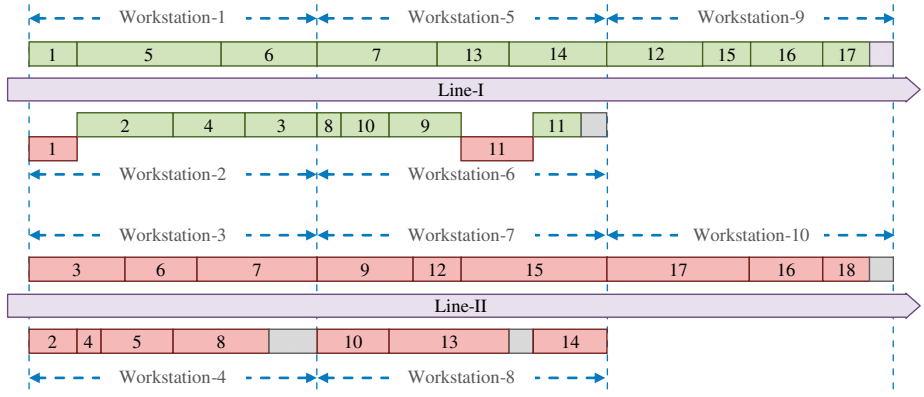


Figure 2. A possible balancing solution using multi-line stations.

ing times of tasks. Dashed areas represent unavoidable idle times caused by capacity and precedence relationships constraints. As it could be seen from the figure, a total of 10 workstations are needed to perform a total of 35 tasks on both of the lines. With the construction of multi-line stations, operator located in workstation-2 on Line-I performs Task-1 from Line-II first, followed by Task-2, Task-4 and Task-3 belonging to Line-I. Similarly, Task-11 from Line-II is completed by workstation-6 located on Line-I. Thus, operators located in workstation-2 and workstation-6 on Line-I contribute to performing tasks on Line-II as well as their main job on Line-I. It should be noted here that Task-14 on Line-II cannot be initialised unless its predecessor task (Task-11) is completed by workstation-6 on the other side of the line. This is one of the most challenging issues in solving PTALBPs and that is why unavoidable idle time occurs before Task-14 on Line-II.

If the lines were balanced individually (without multi-line stations), theoretical minimum number of workstations for Line-I and Line-II could be calculated as $\min K_1 = [53/12]^+ = 5$ and $\min K_2 = [61/12]^+ = 6$, respectively; simply using the well-known formula $\min K_h = [\text{Total Task Time}/\text{Cycle Time}]^+$, where $[X]^+$ denotes the smallest integer greater than or equals to X. Whereas theoretical minimum number of workstations ($\min K$) decreases to 10 ($\min K = [114/12]^+$), one lower than the sum of independent-balancing solutions (5 + 6 = 11), with the opportunity of assigning tasks into a more diversified positions thanks to multi-line stations when the lines are balanced together.

2.2. *Mathematical model*

The notation used in the study can be summarised as below to describe the problem:

2.2.1. *Notation*

h line index ($h = 1, \ldots, H$), where H represents total number of lines,

i task index ($i = 1, \ldots, n_h$), where n_h represents total number of tasks on line h,

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

k station index ($k = 1, \ldots, K$), where K represents total number of utilised workstations

$$X_{hijk} = \begin{cases} 1 & \text{if task } i \text{ is assigned to workstation } k, \text{ on side } j \text{ of line } h \\ 0 & \text{otherwise} \end{cases}.$$

t_{hi} Processing time of task i on line h,

P_h Set of precedence relationships in precedence diagram of line h,

C Common cycle time of the lines,

ZP Set of pairs of tasks that must be assigned to the same workstation, *positive zoning*,

NP Set of pairs of tasks that cannot be assigned to the same workstation, *negative zoning*

t^s_{hi} Starting time of task i on line h,

q_k Queue number that station k is utilised on,

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

$z_k = \{ 1 \quad \text{if station } k \text{ is utilised } 0\text{otherwise.},$

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

σ Variable, ($\sigma = h + 1, \ldots, H$),

β Variable, ($\beta \in \{0, 1\}$),

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

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

$$R_{hrs} = \begin{cases} 1 & \text{if tasks } r \text{ and } s \text{ are assigned to the same workstation on line } h \\ 0 & \text{otherwise} \end{cases}$$

2.2.2. *Objective function*

The objective function used in this study is obtained from the modification of objective functions used in the previous studies (see Chiang (1998) and Ozcan, Gokcen, and Toklu (2010)) and is presented in Equation (1). This nonlinear objective function represents sum of squares of each workstation's workload. So, maximising this objective function helps to reduce the number of stations.

$$\text{Max } Z = \sum_{h=1}^{H} \sum_{j \in \{0,1\}} \sum_{k=1}^{K} \left(\sum_{i=1}^{n_h} t_{hi} X_{hijk} \right)^2 \tag{1}$$

2.2.3. *Constraints*

$$\sum_{k=1}^{K} X_{hijk} = 1, \quad \forall i = 1, \ldots, n_h; \quad \forall h = 1, \ldots, H; \quad \forall j \in \{0, 1\}. \tag{2}$$

$$\sum_{i=1}^{n_h} \left(t_{hi} + t^s_{hi}\right) X_{hijk} + S_k \left(\sum_{i=1}^{n_h} \left(t_{(h+1)i} + t^s_{(h+1)i} \right) X_{(h+1)i(j-1)k} \right) \leq C z_k, \quad \forall k = 1, \ldots, K; \quad \forall h = 1, \ldots, H-1; \quad \forall j \in \{0, 1\}. \tag{3}$$

$$\sum_{i=1}^{n_h} X_{hijk} - n_h U_{hjk} \leq 0, \quad \forall k = 1, \ldots, K; \quad \forall h = 1, \ldots, H; \quad \forall j \in \{0, 1\}. \tag{4}$$

$$|j - 1| \left(U_{h\beta k} + U_{(\sigma-\mu)jk}\right) + j\left(U_{h(j-1+c)k} + U_{\sigma jk}\right) = 1, \quad \forall k = 1, \ldots, K; \quad \forall h = 1, \ldots, H; \quad \forall j \in \{0, 1\}; \quad \forall \sigma = h + 1, \ldots, H; \quad \forall \beta \in \{0, 1\}. \tag{5}$$

$$\sum_{k=1}^{K} q_k \left(X_{hrjk} - X_{hsjk}\right) + R_{hrs} \left(t^s_{hr} + t_{hr} - t^s_{hs}\right) \leq 0, \quad \forall h = 1, \ldots, H; \quad \forall j \in \{0, 1\}; \quad \forall (r, s) \in P_h. \tag{6}$$

$$\sum_{k=1}^{K} X_{hajk} - \sum_{k=1}^{K} X_{hbjk} = 0, \quad \forall (a, b) \in ZP; \quad \forall h = 1, \ldots, H; \quad \forall j \in \{0, 1\}. \tag{7}$$

$$X_{hajk} + X_{hbjk} \leq 1, \quad \forall (a, b) \in ZN; \quad \forall h = 1, \ldots, H; \quad \forall j \in \{0, 1\}; \quad \forall k = 1, \ldots, K. \tag{8}$$

The main objective of the model given in Equation (1) is to minimise the number of workstations by maximising sum of squares of each workstation's workload. Constraint

(2) ensures that all tasks are assigned to a station and each task is assigned only once. Constraint (3) represents cycle time constraint that assures each task is executed before the cycle time. Constraints (4) and (5) ensure that an operator working at station k can perform additional task(s) from only one adjacent line; unless station k is not utilised on left side of the first line or on right side of the last line; i.e. if an operator is located on right side of the first line $(h = 1,\ j = 1)$ that operator can perform additional tasks from only left side of the second line $(h = 2,\ j = 0)$ as well as his/her main job. That operator cannot perform any job from the left side of the first line $(h = 1,\ j = 0)$, or right side of the second line $(h = 2,\ j = 1)$, as it is not possible a direct communication with those tasks assigned to these stations. Please refer to Section 2.2.1 for explanations on variables c and μ given in constraint 5. Constraint (6) ensures that the precedence relationships are not violated and completion times of tasks are considered to avoid interference. Given a task pair $(r, s) \in P_h$, where r is one of the predecessors of s, thgen s can be initialised after r is completed. Constraints (7) and (8) demonstrate positive and negative zoning constraints, respectively. As noted above, ZP is the set of pairs of tasks that must be assigned to the same workstation, while ZN is the set of pairs of tasks that cannot be assigned to the same workstation.

2.3. *Assumptions*

The assumptions considered in the study are as follows:

- Only one product model is assembled on each line, so total number of lines equals to total number of product models.
- Each product model has its own precedence relationships diagram.
- The precedence relationships and task times of each product model are known.
- The operators have no preference about the tasks and workstations.
- Walking times of the operators are ignored.

3. Proposed GA-based approach for PTALBP

GA is an efficient random search algorithm originating from the evolutionary rules of the nature population. Its solution approach is motivated by the biological process of natural selection and the solution of an optimisation problem is encoded as chromosome, where the specific parameters of solution (called genes) are located on the chromosome (Suresh, Vinod, and Sahu 1996). Each individual (chromosome) corresponds to a possible solution and its survival chance through generations is characterised by its fitness value, which is defined in accordance with the objective function. A finite set of individuals constitutes population and usually its size remains fixed through generations. The initial population is built at random and the population is updated by generating new individuals, which replace the old ones, in subsequent iterations. New individuals are created by means of genetic operators, crossover and mutation, and the iterations are terminated when the stopping criterion is satisfied (Borisovsky, Delorme, and Dolgui 2013). The characteristics of the implemented GA approach within the scope of this study are explained below.

3.1. *General outline*

The outline of the proposed GA-based algorithm is exhibited in Figure 3. As can be seen from the figure, the algorithm starts by generating an initial population, which consists of a predefined number (population size) of chromosomes, and continues with the evaluation of created chromosomes. Genetic operators (crossover and mutation) are performed and some completely new chromosomes are also generated randomly with the probability of 2% to keep diversity and avoid early convergence.

After the fitness evaluation of new individuals (resulting from genetic operators and random generation), insufficient chromosomes in the population are replaced with better ones (if any). In the fitness evaluation process, tasks are assigned to the minimum numbered workstations as far as possible. This loop continues until the iteration number is exceeded. Finally, the chromosome which gives the best fitness value is selected as the best solution of the problem.

3.2. *Initial population*

The chromosome is made up of several genes which represent tasks (by the tasks' index numbers) in a sequence. So, each gene of the chromosome is an integer representing a task number of a sequence of tasks to be assigned to the stations. Different solutions and fitness values are examined by changing the order of the genes on the chromosome. Figure 4 represents a sample of task-based chromosome which is used in this study. The length of the chromosome is characterised by the total number of tasks that belongs to the models. If we assume two product models with nine tasks and eight tasks, respectively, gene numbers lower than or equal to nine belong to the Product Model-I (produced on Line-I). The remaining tasks (Task-10–Task-17) belong to the Product Model-II (produced on Line-II) in an incremental order, i.e. Task-10 and Task-13 symbolise Task-1 and Task-4 for Product Model-II.

Initial population is generated randomly using a heuristic algorithm, namely Comsoal (Arcus 1966), to start the GA. But first of all, tasks are grouped according to

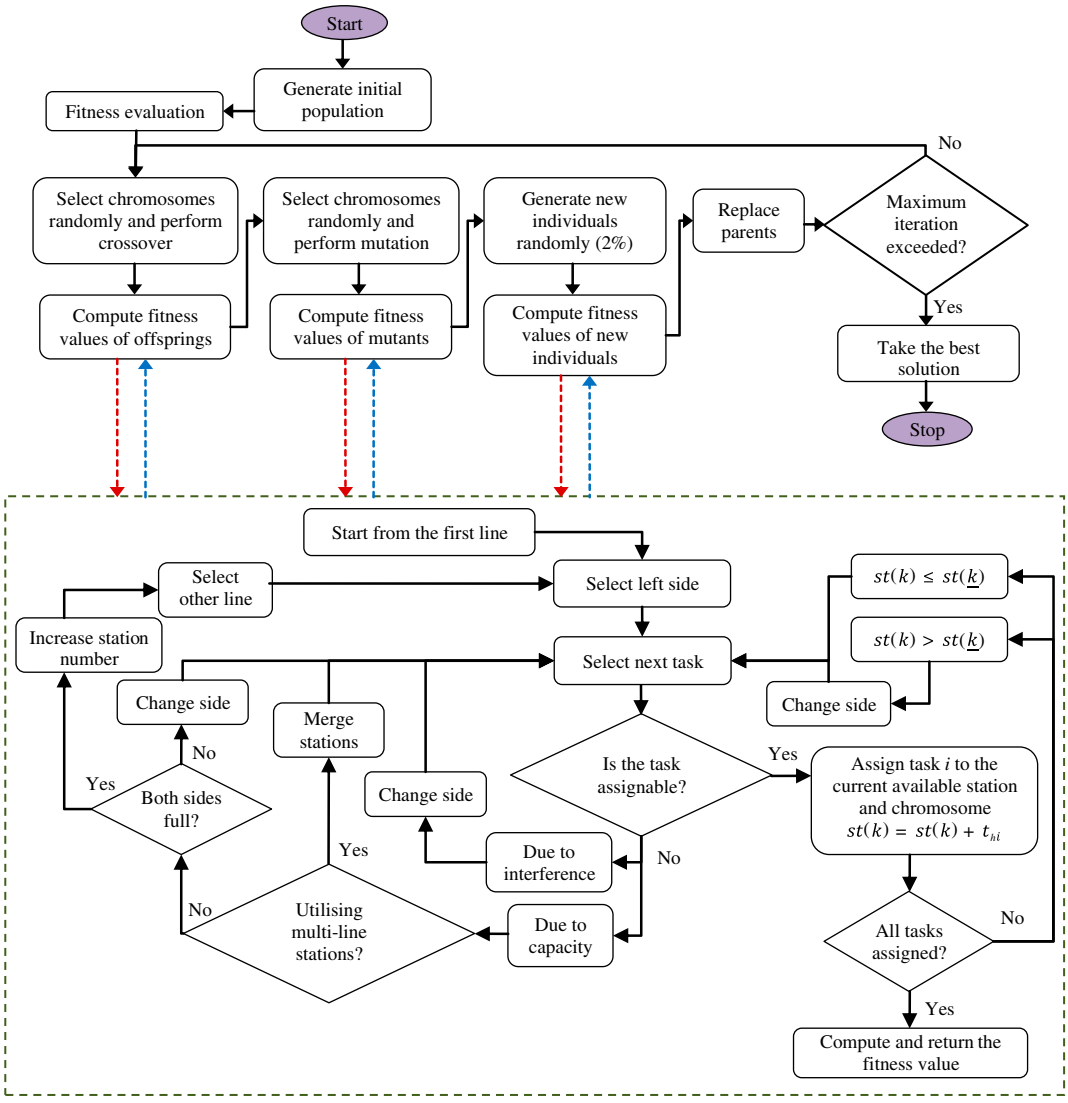


Figure 3. Flowchart of the proposed algorithm, adapted from Kucukkoc and Zhang (2013).

the line and preferred operation direction data. S1LE, S1RE, S2LE and S2RE lists (named S lists) are formed to separate tasks according to their input data. For example, S1LE list consists of tasks that can be assigned to the left side of Line-I. After separating tasks, a Comsoal-based heuristic procedure generates different chromosomes by selecting tasks from these lists until a population is obtained with a predetermined size. This procedure is exhibited in Figure 5.

The process flow as to how a chromosome is generated is also depicted in Figure 5; where available tasks mean those tasks which (i) satisfy capacity constraints of the current station, (ii) have no predecessors or all of their predecessor tasks are already completed and (iii) do not violate interference rule. For each side of each line, tasks with no predecessor and satisfy capacity constraints are selected randomly from relevant list and allocated to the chromosome one by one, then those tasks whose predecessors have been processed and allocated to the chromosome, and so on.

Allocating tasks to the chromosome continues until all tasks are sequenced on the chromosome side by side and line by line. As only the tasks which have no predecessor or whose predecessors have been allocated are

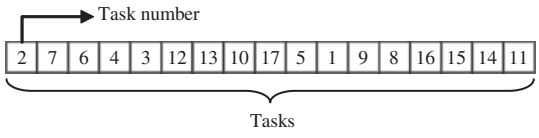


Figure 4. An example of task-based chromosome representation.

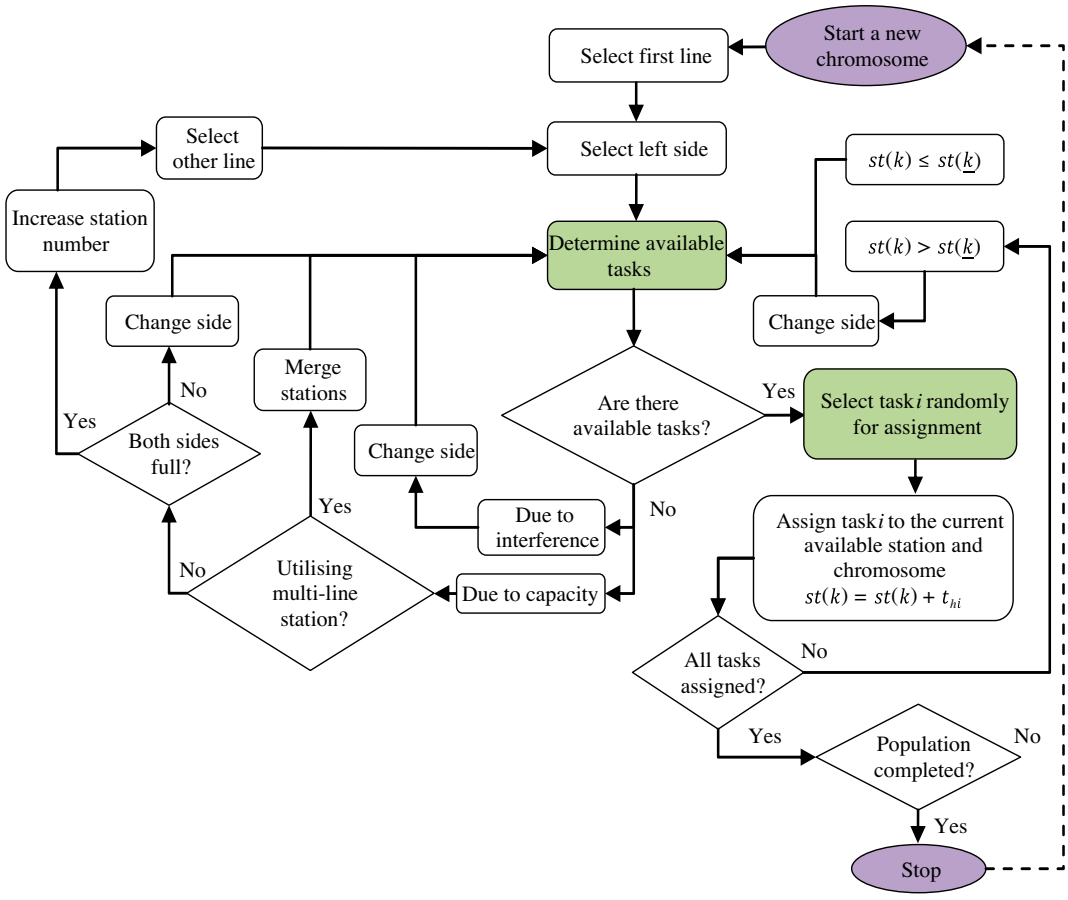


Figure 5. Generating initial population for the proposed algorithm, adapted from Kucukkoc and Zhang (2013).

selected in each loop, infeasible task sequences are naturally filtered out during this process to prevent infeasible chromosomes (solutions) that violate precedence relationships. At this step, the length of each chromosome equals to the total number of tasks on both lines. As can be seen from the figure, if workload of the current station is larger than the workload of its mated station ($st(k) > st(\underline{k})$), then side is changed and candidate tasks for new side are considered. Afterwards, fitness values of the chromosomes are computed (decoding). During the task allocation process, if the current side of a line lies between two lines and there is no available task to be assigned from the current line but from the adjacent line, the multi-line station is utilised so that some tasks can be performed from the other line.

3.3. *Decoding and fitness evaluation*

Decoding procedure is processed by assigning tasks to workstations according to precedence relationships, unless cycle time is not exceeded. The sequence of tasks on the chromosome is considered while allocating tasks to the stations. The initial tasks on the chromosome are assigned in the earliest workstations as far as possible. If the next task in the sequence does not satisfy the precedence or capacity constraints, a new workstation is opened and the task is assigned to this workstation. Fitness value of each chromosome is computed when all tasks are assigned. Total number of workstations are also recorded for each chromosome in order to compare the obtained results with previous tabu search algorithm proposed by Ozcan, Gokcen, and Toklu (2010).

3.4. *Selection, crossover and mutation*

Crossover operator takes two parent individuals and produces two offspring by combining and exchanging their elements. Roulette wheel (Baker 1987) is used to select parent chromosomes from the population to keep diversity and avoid local minima. Two-point crossover operator is applied to recombine the chromosomes. During this process, infeasible solutions are not allowed since missing parts of both offspring are built according to precedence relationships. Selected two parents are divided into three sections: head, middle and tail. Cutting points, which cut each parent into three parts, are

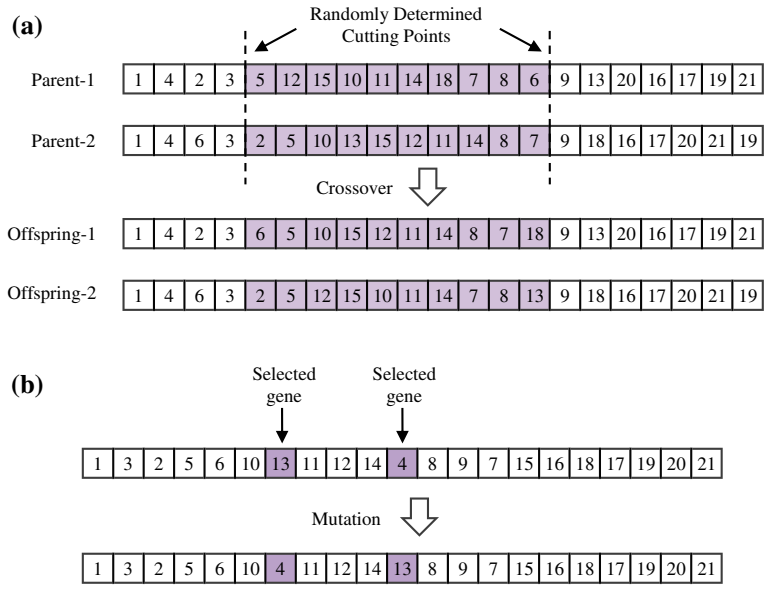


Figure 6. Illustration of (a) crossover and (b) mutation procedures.

determined randomly for each parent pair. By this way, diversity is preserved in the population and it is enabled to search the solution space effectively. First offspring keeps the head and tail parts of the first parent and the middle part of the first offspring is filled by adding missing tasks according to the order in which they are contained in the second parent. Similarly, second offspring is formed by head part of the second parent, missing tasks according to the order in which they are contained in the first parent, and tail part of the second parent (Leu, Matheson, and Rees 1994; Akpinar and Bayhan 2011). An example of the crossover procedure used in this study is given in Figure 6(a). As could be seen from the figure, missing tasks of Offspring-1 are 5, 12, 15, 10, 11, 14, 18, 7, 8 and 6. These tasks appear in the sequence of 6, 5, 10, 15, 12, 11, 14, 8, 7 and 18 on Parent-2, and constitute the middle part of Offspring-1.

Mutation is applied to add random changes to an individual and it plays a critical role in GA to keep diversity by changing the order of the genes dramatically. Roulette wheel selection strategy is applied so that an individual with a high fitness value will have more chance to be chosen as a parent than the ones with a lower fitness. To mutate a chromosome, two genes are selected randomly and swapped by considering precedence relationships among tasks. An example of mutation procedure is presented in Figure 6(b).

3.5. *Forming new generation*

New generation is formed by comparing the fitness values of new individuals, which are obtained from crossover – mutation procedures and random generations, with existing chromosomes in the population and replacing the worst chromosomes in the population with better ones (if any).

4. Illustrative example

To explain the running mechanism of the proposed algorithm and the encoding-decoding procedures in particular, a numerical example is given in this section. In the example, meaning of the genes, decoding procedure of the tasks and assigning tasks to the stations can be investigated visually.

Two well-known test problems, P9 and P12 (Kim, Kim, and Kim 2000), are taken from the literature and given in Figure 7 to be considered as precedence relationships and task processing times of two different product models (Product Model-I and Product Model-II respectively). Numbers in nodes represent task numbers while arrows between nodes symbolise precedence relationships between tasks. To make encoding-decoding procedures easier, each task number given in nodes and belonging to Product Model-II (P12) is represented by the sum of the total number of tasks belonging to Product Model-I (P9) and the original task number. For example, Task-5 of Product Model-II is represented as 14 (5 + 9); and Task-11 is represented as 20 (11 + 9). So, $n_1 + n_2 = 12 + 9 = 21$ tasks are subject to balancing. Processing time and preferred operation direction of each task, which represents the side where tasks can be assigned, are also given over each node (L, R and E denote left, right and either sides, respectively). S lists, which represent candidate tasks that can be allocated to the relevant side of each line, can be constructed as in Figure 8.

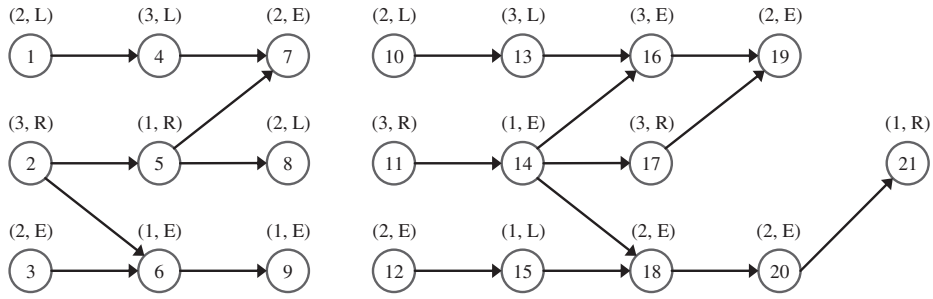


Figure 7. Precedence diagrams for the illustrative example: (a) P9, and (b) P12, adapted from Kim, Kim, and Kim (2000).

Task assigning process to the chromosome and workstations can be seen for the first 12 steps from Figures 9 and 10, respectively. The task allocation process starts from the left side of the Line-I. If the cycle time is assumed six time units for both lines, available tasks for this particular side are Task-1 and Task-3, because Task-4–Task-9 have predecessors which have not been completed yet. Task-3 is selected randomly, and allocated to the chromosome (Step 1 in Figure 9) and to the current available station that has enough capacity (see Figure 10). Then, the station time of the current workstation is increased by the amount of assigned task's processing time. As the station time of the current workstation is larger than the station time of its mated workstation ($st(k) > st(\underline{k})$), the side is changed and available tasks are determined for the new side again. One of the available tasks (Task-2) is selected and allocated to the chromosome and to the current workstation. In Step 3, Task-6 can be initialised after completion of its predecessors, Task-2 and Task-3 (to avoid interference). When both sides of the current line are full or there is no enough capacity for assignment, the line is changed and available tasks are allocated to the adjacent line concurrently with the chromosome. This cycle continues until all tasks are assigned to the chromosome.

Line I
Left Side $S1LE = \{1, 3, 4, 6, 7, 8, 9\}$
Right Side $S1RE = \{2, 3, 5, 6, 7, 9\}$

Line II
Left Side $S2LE = \{10, 12, 13, 14, 15, 16, 18, 19, 20\}$
Right Side $S2RE = \{11, 12, 14, 16, 17, 18, 19, 20, 21\}$

Figure 8. S lists are shown on parallel two-sided assembly lines.

Fitness evaluation of the new individuals obtained from the genetic operators and random generations are computed after the chromosome is decoded. To illustrate the decoding process, an example of decoded chromosome is demonstrated below. In the example, decoding procedure of the tasks and assigning tasks to the workstations can be investigated easily. The example chromosome given in Figure 11(a) can be decoded as in Figure 11(b) under a cycle time constraint of six time units.

Step No	Line-Side	Available Tasks	Selected Task	Chromosome
1	1-L	1, 3	3	3
2	1-R	2, 6	2	3, 2
3	1-L	1, 6	6	3, 2, 6
4	1-R	5, 9	9	3, 2, 6, 9
5	1-L	1	1	3, 2, 6, 9, 1
6	1-R	5	5	3, 2, 6, 9, 1, 5
7	2-L	10, 12	10	3, 2, 6, 9, 1, 5, 10
8	2-R	11, 12	11	3, 2, 6, 9, 1, 5, 10, 11
9	2-L	12, 13, 14	13	3, 2, 6, 9, 1, 5, 10, 11, 13
10	2-R	12, 14	12	3, 2, 6, 9, 1, 5, 10, 11, 13, 12
11	2-R	14	14	3, 2, 6, 9, 1, 5, 10, 11, 13, 12, 14
12	2-L	15	15	3, 2, 6, 9, 1, 5, 10, 11, 13, 12, 14, 15
…	…	…	…	…

Figure 9. Task sequencing process to the chromosome.

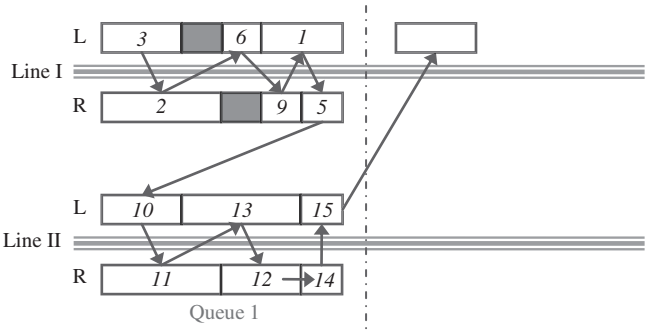


Figure 10. Task allocation process to the workstations.

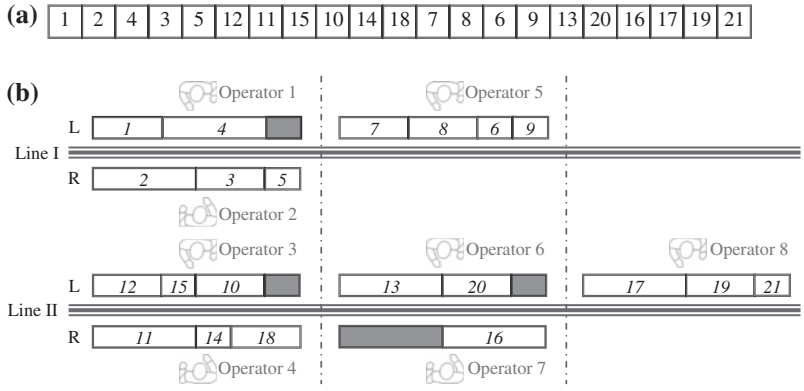


Figure 11. (a) A chromosome sample and (b) its decoded line configuration.

As can be seen in Figure 11(b), eight operators are needed to assemble two different product models for the given example. On the right side of the Line-II, Task-16 can be initialised upon its predecessor task, Task-13, is completed on the left side of the line.

5. Computational study

The proposed algorithm was coded in Java SE 7u4 environment and run on a 3.1 GHz Intel Core i5-2400 CPU 4 GB RAM computer to test the performance of the proposed algorithm solving the test problems originally combined by Ozcan, Gokcen, and Toklu (2010). Seven original well-known problems from the literature: P9, P12 and P24 from Kim, Kim, and Kim (2000); P16, A65 and A205 from Lee, Kim, and Kim (2001); and B148 from Bartholdi (1993) (B148 was then modified by Lee, Kim, and Kim (2001)) were derived by Ozcan, Gokcen, and Toklu (2010) in different combinations to test the performance of tabu search algorithm in solving PTALBPs. In order to analyse the efficiency of the proposed approach in the current research, these test problems are solved using the proposed GA in two stages. In the first stage, the problems are solved using the cycle times provided by (Ozcan, Gokcen, and Toklu 2010) and the obtained results are compared with the results of Ozcan, Gokcen, and Toklu (2010) to have an idea about the overall performance of the proposed GA. In the second stage, test problems are solved for binary combinations of different cycle time values of Line-I and Line-II. The main objectives are (i) to observe the response of the entire system to different levels of the parallel lines' cycle times and (ii) to determine the best cycle time pair which gives the highest line efficiency.

5.1. *Stage-1: comparison with the existing results*

The algorithm is run using the parameters given in Table 2 to solve the test problems given in Table 3 and the best solution is taken after three runs for each test problem. As could be seen from Table 2, used parameters may differ from one test problem to another in order to scan search space more effectively and increase the solution building capacity of the algorithm, especially in the large-sized problems, as the search space grows exponentially with the increasing number of tasks. These parameters are chosen experimentally for a high-quality solution in an acceptable period of time. Table 3 presents

Table 2. Parameters of the proposed GA.

Test problem	Population size	Crossover rate	Mutation rate	Number of iterations
1–8	20	0.2	0.10	30
9–12	30	0.3	0.15	40
13–16	40	0.3	0.15	60
17–22	50	0.4	0.20	100
23–28	60	0.4	0.20	150
29–32	80	0.5	0.20	200

Table 3. Data for test problems.

Problem No	Test problems (Line I–Line II)	Number of tasks (Line I–Line II)	Cycle time (Line I–Line II)
1	P9–P9	9–9	3–3
2	P9–P9	9–9	4–5
3	P9–P12	9–12	6–6
4	P9–P12	9–12	4–7
5	P12–P12	12–12	5–5
6	P12–P12	12–12	6–7
7	P12–P16	12–16	7–16
8	P12–P16	12–16	8–21
9	P16–P16	16–16	16–16
10	P16–P16	16–16	19–21
11	P16–P24	16–24	19–35
12	P16–P24	16–24	22–40
13	P24–P24	24–24	18–18
14	P24–P24	24–24	20–24
15	P24–A65	24–65	30–490
16	P24–A65	24–65	20–544
17	A65–A65	65–65	381–381
18	A65–A65	65–65	435–435
19	A65–A65	65–65	490–544
20	A65–B148	65–148	381–408
21	A65–B148	65–148	490–459
22	A65–B148	65–148	544–510
23	B148–B148	148–148	408–408
24	B148–B148	148–148	306–357
25	B148–B148	148–148	459–510
26	B148–A205	148–205	306–1888
27	B148–A205	148–205	510–2832
28	B148–A205	148–205	255–1510
29	A205–A205	205–205	1510–1510
30	A205–A205	205–205	2832–2832
31	A205–A205	205–205	2077–2266
32	A205–A205	205–205	2454–2643

the test problems used by Ozcan, Gokcen, and Toklu (2010) along with the cycle times considered when solving these problems in the first stage of the experimental tests.

Table 4 exhibits the results (number of workstations) obtained using the proposed GA for each test problem under designated cycle time constraints. The number of stations obtained from the proposed algorithm is compared with the independent line balance of the two-sided assembly lines, the theoretical minimum number of stations (LB) and the tabu search algorithm proposed by Ozcan, Gokcen, and Toklu (2010) (which is the only study available in the literature and given as TS in Table 4). Obtained results are compared with respect to total number of required workstations as this is the only result reported by Ozcan, Gokcen, and Toklu (2010).

Table 4. Comparison of the obtained computational results by means of total number of required workstations.

			Balancing together	
Problem no	Independent balancing (Line 1 + Line 2)	Theoretical minimum number of stations (LB)	TS (Ozcan, Gokcen, and Toklu 2010b)	Proposed GA
1	6 + 6	12	12	12
2	5 + 4	8	8	8
3	3 + 5	7	8	8
4	5 + 4	8	9	9
5	6 + 6	10	11	11
6	5 + 4	8	9	9
7	4 + 6	9	10	10
8	4 + 5	8	8	8
9	6 + 6	11	11	11
10	5 + 5	9	10	10
11	5 + 4	9	9	9
12	4 + 4	8	8	8
13	8 + 8	16	16	16
14	8 + 6	13	14	14
15	5 + 11	16	16	16
16	8 + 10	17	18	18
17	15 + 15	27	29	29
18	13 + 13	24	25	25
19	11 + 10	20	21	21
20	15 + 13	26	28	28
21	11 + 12	22	23	23
22	10 + 11	20	21	21
23	13 + 13	26	26	26
24	18 + 15	32	33	33
25	12 + 11	22	23	23
26	18 + 15	30	33	33
27	11 + 10	19	21	21
28	21 + 18	36	39	38
29	18 + 18	31	36	35
30	10 + 10	17	20	20
31	14 + 12	22	26	26
32	12 + 11	19	23	23

Theoretical minimum number of workstations is calculated by Ozcan, Gokcen, and Toklu (2010). They modified simple lower bound equation proposed by Hu, Wu, and Jin (2008) for the two-sided assembly line balancing problems. As the calculation of lower bound does not take the precedence constraints (Akpinar and Bayhan 2011) into consideration, real value of the lower bound is most likely larger than the computed value, and this situation must be taken into account to measure the efficiency of the developed approach and comparison with the LB. Since the optimal number of stations cannot be less than LB, if the obtained number of stations equals the LB, then it can be said that the obtained result is optimal (Ozcan, Gokcen, and Toklu 2010). As can be seen from Table 4, the proposed GA discovered optimal

solutions for nine of the 32 test problems. Moreover, GA produced one less workstation than the tabu search algorithm for the test problems 28 and 29. Therefore, it could be said that the proposed GA-based approach has a promising solution capacity for the PTALBPs.

A paired two-samples *t*-Test is conducted using data analysis tool available in Microsoft Excel™ 2010 to determine whether there is a significant difference between *independent balancing* and *together balancing* of the lines in terms of the means of number of workstations needed. The results presented in *independent balancing* and *proposed GA* columns in Table 4 are subject to consideration for this statistical test. The null and alternative hypotheses are stated at the $a = 0.05$ level (95%) for means of workstation numbers obtained when the lines are balanced independently (μ_I) and when the lines are balanced together using GA (μ_T) as follows:

H_0: There is no significant difference between the means of workstation numbers obtained by the solution strategies in favour of the alternative ($\mu_I \leq \mu_T$).

H_1: Balancing lines together using the proposed GA approach significantly reduces the number of workstations needed ($\mu_I > \mu_T$).

As seen from the hypotheses, the test is designed as one-tailed. The summary of the test results given in Table 5 (see Test-1 column) indicates that there is significant difference in the means of workstation numbers when the lines are balanced independently $(\mu_I = 19.06, Var_I = 83.42)$ and when the lines are balanced together using the proposed GA-based approach $(\mu_T = 18.81, Var_T = 82.29)$; $t(31) = 3.2146$, $p = 0.0015$. Thus, the null hypothesis is rejected with very strong evidence. These results suggest that balancing lines together allowing multi-line stations helps reduce total number of required workstations significantly.

Using the results proposed in the relevant columns of Table 4, another paired two-samples *t*-Test is performed to determine whether there is a significant difference between the means of workstation numbers found by TS (Ozcan, Gokcen, and Toklu 2010) and proposed GA. The null and alternative hypotheses stated at the $a = 0.05$ level (95%) for means of workstation numbers obtained using TS (μ_{TS}) and GA (μ_{GA}) are as follows:

H_0: There is no significant difference between the means of workstation numbers obtained by the solution strategies in favour of the alternative ($\mu_{TS} \leq \mu_{GA}$).

H_1: GA algorithm finds better solutions than TS when balancing parallel two-sided assembly lines ($\mu_{TS} > \mu_{GA}$).

Based on the summary of the test results presented in Table 5 (see *Test-2* column), there is no strong evidence to reject the null hypothesis at $a = 0.05$. Therefore, it is not possible to argue that GA $(\mu_{GA} = 18.81, Var_{GA} = 82.29)$ performs significantly better than TS $(\mu_{TS} = 18.88, Var_{TS} = 84.63)$ at this confidence level; $t(31) = 1.4376, p = 0.08$. However, it could be argued that GA finds significantly better solutions than TS if the test was performed at $a = 0.1$, and it can be clearly seen from Table 4 that GA finds quite promising results.

5.2. *Stage-2: solutions for various cycle time situations*

Now, we can proceed to the second stage of the computational tests assuming that the performance of the proposed GA-based algorithm is sufficient enough. In this stage, the test problems given above are solved using the proposed GA (with the same GA parameters used in the previous subsection) by considering different cycle time for the lines. Four levels are determined for cycle time of each line in each test problem and the problems are solved under the constraints of these cycle time combinations. Considered cycle time for Line-I and Line-II, calculated common cycle time (C), and obtained number of stations (K) are given in Table 6 for different test cases.

The *LE* column reports the computed system efficiency based on the obtained number of workstations. This value is obtained via dividing total needed time to

Table 5. Results of the paired two-samples *t*-test for means of workstation numbers.

	Test-1		Test-2	
Paired two-samples *t*-test	Independent balancing	Together balancing (GA)	Together balancing (TS)	Together balancing (GA)
Mean (μ)	19.06	18.81	18.88	18.81
Variance (Var)	83.42	82.29	84.63	82.29
Observations	32	32	32	32
Pearson correlation	0.998		0.999	
Hypothesised mean difference	0		0	
Degrees of freedom	31		31	
t Stat	3.2146		1.4376	
$p(T \leq t)$ one-tail	0.0015		0.0803	
t critical one-tail	1.6955		1.6955	
$p(T \leq t)$ two-tail	0.0030		0.1606	
t critical two-tail	2.0395		2.0395	

Table 6. Computational results for different cycle time levels of the lines.

P9–P9

T. Task Times		Cycle time of Line-II											
P9	P9	4			5			6			7		
17	17	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	3	12	11	0.90	15	10	0.91	6	9	0.94	21	9	0.90
	4	4	9	0.94	20	9	0.85	12	8	0.89	28	8	0.83
	5	**20**	**8**	**0.96**	5	8	0.85	30	7	0.89	35	7	0.83
	6	12	8	0.89	30	7	0.89	6	6	0.94	42	6	0.88

P9–P12

T. Task Times		Cycle time of Line-II											
P9	P12	6			8			10			12		
17	25	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	4	12	10	0.84	**8**	**8**	**0.92**	20	8	0.84	12	7	0.90
	6	6	8	0.88	24	7	0.85	30	6	0.89	12	6	0.82
	8	24	8	0.79	8	6	0.88	40	6	0.77	24	6	0.70
	10	30	7	0.84	40	6	0.80	10	5	0.84	60	5	0.76

P12–P12

T. Task Times		Cycle time of Line-II											
P12	P12	6			8			10			12		
25	25	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	5	30	11	0.83	40	10	0.81	10	9	0.83	**60**	**8**	**0.89**
	7	42	9	0.86	56	8	0.84	70	7	0.87	84	7	0.81
	9	18	8	0.87	72	7	0.84	90	7	0.75	36	6	0.81
	11	66	8	0.80	88	7	0.77	110	6	0.80	132	6	0.73

P12–P16

T. Task Times		Cycle time of Line-II											
P12	P16	16			18			20			22		
25	82	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	8	16	10	0.83	72	9	0.85	40	9	0.80	88	8	0.86
	10	80	9	0.85	90	9	0.78	20	8	0.83	110	7	0.89
	12	48	9	0.80	36	8	0.83	60	8	0.77	132	7	0.83
	14	112	8	0.86	126	8	0.79	140	8	0.74	**154**	**6**	**0.92**

P16–P16

T. Task Times		Cycle time of Line-II											
P16	P16	17			19			21			23		
82	82	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	14	238	13	0.82	266	13	0.78	42	13	0.75	322	12	0.79
	16	272	12	0.83	304	12	0.79	336	12	0.75	**368**	**10**	**0.87**
	18	306	12	0.78	342	12	0.74	126	11	0.77	414	11	0.74
	20	340	11	0.81	380	11	0.77	420	11	0.73	460	10	0.77

(*Continued*)

Table 6. (*Continued*).

P16–P24

T. Task Times		Cycle time of Line-II											
P16	P24	25			27			29			31		
82	140	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	20	100	11	0.88	540	11	0.84	580	11	0.81	620	11	0.78
	22	550	11	0.85	594	10	0.89	638	10	0.86	682	10	0.82
	24	**600**	**10**	**0.90**	216	10	0.86	696	10	0.82	744	10	0.79
	26	650	10	0.88	702	10	0.83	754	10	0.80	806	10	0.77

P24–P24

T. Task Times		Cycle time of Line-II											
P24	P24	19			21			23			25		
140	140	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	18	342	17	0.89	126	16	0.90	**414**	**15**	**0.92**	450	15	0.89
	20	380	16	0.90	420	16	0.85	460	15	0.87	100	15	0.84
	22	418	15	0.92	462	15	0.87	506	15	0.83	550	14	0.85
	24	456	15	0.88	168	14	0.89	552	14	0.85	600	13	0.88

A65–A65

T. Task Times		Cycle time of Line-II											
A65	A65	385			425			465			505		
5099	5099	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	360	27,720	32	0.86	30,600	31	0.84	11,160	29	0.87	36,360	28	0.87
	390	30,030	30	0.88	33,150	28	0.90	12,090	28	0.86	39,390	27	0.86
	420	4620	29	0.88	35,700	28	0.86	13,020	27	0.86	42,420	26	0.86
	450	34,650	29	0.85	**7650**	**26**	**0.90**	13,950	26	0.86	45,450	25	0.86

A65–B148

T. Task Times		Cycle time of Line-II											
A65	B148	375			400			425			450		
5099	5024	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	360	9000	32	0.86	3600	31	0.86	**30,600**	**30**	**0.87**	1800	30	0.84
	380	28,500	32	0.84	**7600**	**30**	**0.87**	**32,300**	**29**	**0.87**	17,100	29	0.85
	400	6000	31	0.84	400	30	0.84	6800	29	0.85	3600	28	0.85
	420	10,500	30	0.85	8400	29	0.85	35,700	28	0.86	6300	28	0.83

(*Continued*)

Table 6. (*Continued*).

B148–B148

T. Task Times		Cycle time of Line-II											
B148	B148	300			350			400			450		
5024	5024	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	325	3900	38	0.85	4550	34	0.88	5200	33	0.85	5850	32	0.83
	375	**1500**	**34**	**0.89**	5250	32	0.87	6000	30	0.87	2250	30	0.82
	425	5100	34	0.84	5950	31	0.84	6800	29	0.84	7650	28	0.82
	475	5700	33	0.83	6650	29	0.86	7600	28	0.83	8550	26	0.84

A205–A205

T. Task Times		Cycle time of Line-II											
A205	A205	1550			1850			2150			2450		
23,345	23,345	C	K	LE	C	K	LE	C	K	LE	C	K	LE
Cycle time of Line-I	1475	91,450	40	0.77	**109,150**	**36**	**0.79**	126,850	34	0.78	144,550	34	0.75
	1850	57,350	36	0.77	**1850**	**32**	**0.79**	79,550	32	0.73	90,650	30	0.74
	2225	137,950	34	0.75	164,650	31	0.75	191,350	28	0.76	218,050	27	0.74
	2600	80,600	32	0.75	96,200	29	0.74	111,800	27	0.73	127,400	25	0.74

Note: 'T. Task Times' column gives the sum of all task times for the relevant problem. The bold values represent the best (maximum) LE value for the considered problems.

perform all tasks on the lines by the total available time of the utilised system (see Equation 9, the definitions of the used symbols have already been given in Section 2.2).

$$LE = \frac{\sum_{h=1}^{H} \frac{C}{C_h} \sum_{i=1}^{n_h} t_{hi}}{K \times C}. \quad (9)$$

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

Although the optimality of the solutions cannot be guaranteed, our findings indicate that near-optimal solutions can be obtained very quickly for even large-sized instances. As could be seen from the obtained results, different solutions are obtained with different line efficiency values corresponding to the binary combinations of the cycle time levels of the parallel lines for the same test problem. The proposed algorithm finds high-quality solutions over 85% efficiency for the entire test problems studied, except the case A205–A205. The best obtained efficiency result (96%) belongs to the problem P9–P9 and is obtained with the cycle time combination of 5 and 4 for Line-I and Line-II, respectively. The next best result, 92%, is obtained for the problems P9–P12, P12–P16 and P24–P24. As a result of the NP-hard characteristic of the studied problem, it is reasonable that the efficiency value obtained for the largest problem (79% for A205–A205) is lower than those for the small-sized ones as expected. However, even this result could be quite reasonable for real world scenarios.

Obtained efficiency values across two dimensions (the cycle time of Line-I and the cycle time of Line-II) are also plotted as a surface graph and are depicted in the Appendix 1. Thus, readers can see the best provided cycle time combination, which gives the highest efficiency, for each case easily. Also, managers can pick up the *number of workstation – cycle time* combination that fits their organisations and model demands from the provided results.

6. Conclusions

The simple assembly line balancing problem is an NP-hard class of combinatorial problem, as shown by Wee and Magazine (1982). Since the PTALBP is a much more complex version of the simple assembly line balancing problem, it is also NP-hard. The solution space

grows exponentially as the number of tasks increases, which means that obtaining an optimal solution when the problem size increases is very difficult (Kalayci and Gupta 2014). It is the major reason why a considerable amount of researches in the literature strives to develop heuristics and meta-heuristics instead of exact methods to solve the assembly line balancing problems.

We developed the first mathematical model for a recently introduced production planning problem, *PTALBP*, and proposed an alternative possible approach for the solution of the problem. Since the problem is very complex and the size of the problems that can be solved in an acceptable amount of time is drastically limited, we applied GA to the PTALBP, which is the first GA-based approach to solve such a problem, and have obtained very encouraging results. To assess the performance of the algorithm, a set of test problems, previously combined and solved by Ozcan, Gokcen, and Toklu (2010), are solved and obtained results are compared with Ozcan, Gokcen, and Toklu (2010). Although the complexity of the problem is higher than other configurations of assembly lines (i.e. one-sided straight assembly lines), computational results demonstrate that the performance of the proposed algorithm is sufficient. Moreover, the effect of different cycle time situations on the efficiency of the overall line system is also studied for a parallel line system for the first time in the literature. For this aim, parallel lines are balanced for different combinations of their different cycle time levels and the efficiency of the entire line system is reported along with the total number of required stations for each case. Obtained line efficiencies are plotted as surface charts across cycle times of the parallel lines to make analyses easier. Thus, line managers can easily pick up the *number of workstations – cycle time* pair that suits their company best.

This study makes it clear that more research is needed to fill in the gap in the literature on minimising cycle time of the parallel two-sided lines as well as total number of required workstations. Hybrid meta-heuristics and/or hyper-heuristics might also be proposed to increase the solution capacity of the algorithm; or exact solution procedures may be developed to solve the PTALBP, even not the large-sized instances. In addition, workload smoothness between workstations and lines may be of interest for future studies with some more realistic conditions of real applications (i.e. zoning constraints, task synchronisation constraints, positional constraints, etc.).

Acknowledgement

The first author gratefully acknowledges the financial support from the *Balikesir University* and the *Turkish Council of Higher Education* during his PhD at the *University of Exeter* in England. Both authors are grateful to anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

Notes on contributors

Ibrahim Kucukkoc is a PhD candidate in Department of Manufacturing Engineering at University of Exeter, UK, funded by a 4-year scholarship from Balikesir University and Turkish Council of Higher Education. He has also worked as a Research Assistant at Balikesir University, Department of Industrial Engineering, since December 2009. He has published in esteemed refereed international journals including *International Journal of Production Economics* and *International Journal of Production Research*; served as a member of editorial board and as a reviewer for many journals. He has also been involved in organisation of international conferences, such as OR55 and YOR18. His areas of particular interest include assembly line balancing, production planning, operations research, heuristic algorithms, genetic algorithm, ant colony optimisation, multi-agent systems, and artificial intelligence.

David Z. Zhang is a full Professor of Manufacturing Systems at University of Exeter, UK; currently, he is also Head of Advanced Technologies Research Institute and Director of Exeter Manufacturing and Enterprise Centre (XMEC). Professor Zhang's current research focuses on Innovation Road-mapping, Design for Supply Chain Optimisation, Modelling of Consumer Dynamics, Complexity Science (Modelling and Simulation of Complex Systems Involving Socially Interacting Elements), and Dynamically Integrated Product/Manufacturing Systems. He has published over 120 articles in refereed international journals and conference proceedings. His research in the last 10 years has been supported by over £3 million research grants from the EPSRC, EU, DTI and industry. Professor Zhang is a Senior Member of IEEE, and a Member of Technical Committee of IASTED.

ORCID

Ibrahim Kucukkoc http://orcid.org/0000-0001-6042-6896
David Z. Zhang http://orcid.org/0000-0002-1561-0923

References

Akpinar, S., and G. M. Bayhan. 2011. "A Hybrid Genetic Algorithm for Mixed Model Assembly Line Balancing Problem with Parallel Workstations and Zoning Constraints." *Engineering Applications of Artificial Intelligence* 24 (3): 449–457.

Arcus, A. 1966. "COMSOAL, a Computer Method of Sequencing Operations for Assembly Lines." *The International Journal of Production Research* 4 (4): 259–277.

Baker, J. E. 1987. "Reducing Bias and Inefficiency in the Selection Algorithm." In *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, 14–21. Hillsdale, New Jersey, USA.

Bartholdi, J. J. 1993. "Balancing Two-sided Assembly Lines: A Case Study." *International Journal of Production Research* 31 (10): 2447–2461.

Baykasoglu, A., and T. Dereli. 2008. "Two-sided Assembly Line Balancing Using an Ant-colony-based Heuristic." *The International Journal of Advanced Manufacturing Technology* 36 (5–6): 582–588.

Baykasoglu, A., L. Ozbakir, L. Gorkemli, and B. Gorkemli. 2009. "Balancing Parallel Assembly Lines via Ant Colony Optimization." *Cie: 2009 International Conference on Computers and Industrial Engineering* 1–3: 506–511.

Borisovsky, P. A., X. Delorme, and A. Dolgui. 2013. "Genetic Algorithm for Balancing Reconfigurable Machining Lines." *Computers & Industrial Engineering* 66 (3): 541–547.

Buyukozkan, K., I. Kucukkoc, and D. Z. Zhang. 2014. "Lexicographic Bottleneck Mixed-model Assembly Line Balancing Problem: An Artificial Bee Colony Approach." In *Proceedings of the 44th International Conference on Computers and Industrial Engineering (CIE44)*, Istanbul, 1213–1227. October 14–16.

Cercioglu, H., U. Ozcan, H. Gokcen, and B. Toklu. 2009. "A Simulated Annealing Approach for Parallel Assembly Line Balancing Problem." *Journal of the Faculty of Engineering and Architecture of Gazi University* 24 (2): 331–341.

Chiang, W. C. 1998. "The Application of a Tabu Search Metaheuristic to the Assembly Line Balancing Problem." *Annals of Operations Research* 77 (1998): 209–227.

Chica, M., O. Cordon, and S. Damas. 2011. "An Advanced Multiobjective Genetic Algorithm Design for the Time and Space Assembly Line Balancing Problem." *Computers & Industrial Engineering* 61 (1): 103–117.

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.

Gökçen, H., K. Agpak, and R. Benzer. 2006. "Balancing of Parallel Assembly Lines." *International Journal of Production Economics* 103 (2): 600–609.

Goncalves, J. F., and J. R. de Almeida. 2002. "A Hybrid Genetic Algorithm for Assembly Line Balancing." *Journal of Heuristics* 8 (6): 629–642.

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.

Hu, X. F., E. F. Wu, J. S. Bao, and Y. Jin. 2010. "A Branch-and-bound Algorithm to Minimize the Line Length of a Two-sided Assembly Line." *European Journal of Operational Research* 206 (3): 703–707.

Hu, X. F., E. F. Wu, and Y. Jin. 2008. "A Station-oriented Enumerative Algorithm for Two-sided Assembly Line Balancing." *European Journal of Operational Research* 186 (1): 435–440.

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.

Kalayci, Can B., and Surendra M. Gupta. 2014. "A Tabu Search Algorithm for Balancing a Sequence-dependent Disassembly Line." *Production Planning & Control* 25 (2): 149–160.

Kara, Y., H. Gokcen, and Y. Atasagun. 2010. "Balancing Parallel Assembly Lines with Precise and Fuzzy Goals." *International Journal of Production Research* 48 (6): 1685–1703.

Khorasanian, D., S. R. Hejazi, and G. Moslehi. 2013. "Two-sided Assembly Line Balancing Considering the Relationships between Tasks." *Computers & Industrial Engineering* 66 (4): 1096–1105.

Kim, Y. K., Y. H. Kim, and Y. J. Kim. 2000. "Two-sided Assembly Line Balancing: A Genetic Algorithm Approach." *Production Planning & Control* 11 (1): 44–53.

Kim, Y. K., W. S. Song, and J. H. Kim. 2009. "A Mathematical Model and a Genetic Algorithm for Two-sided Assembly Line Balancing." *Computers & Operations Research* 36 (3): 853–865.

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.

Kucukkoc, I., and D. Z. Zhang. 2013. "Balancing Parallel Two-sided Assembly Lines via a Genetic Algorithm Based Approach." *Proceedings of the 43rd International Conference on Computers and Industrial Engineering (CIE43)*, 1–16. Hong Kong: The University of Hong Kong.

Kucukkoc, I., and D. Z. Zhang. 2014a. "An Agent Based Ant Colony Optimisation Approach for Mixed-model Parallel Two-sided Assembly Line Balancing Problem." In *Pre-prints of the Eighteenth International Working Seminar on Production Economics*, 313–328. Innsbruck, Austria, February 24–28.

Kucukkoc, I., and D. Z. Zhang. 2014b. "Mathematical Model and Agent Based Solution Approach for the Simultaneous Balancing and Sequencing of Mixed-model Parallel Two-sided Assembly Lines." *International Journal of Production Economics* 158: 314–333.

Kucukkoc, I., and D. Z. Zhang. 2014c. "Simultaneous Balancing and Sequencing of Mixed-model Parallel Two-sided Assembly Lines." *International Journal of Production Research* 52 (12): 3665–3687.

Lee, T. O., Y. Kim, and Y. K. Kim. 2001. "Two-sided Assembly Line Balancing to Maximize Work Relatedness and Slackness." *Computers & Industrial Engineering* 40 (3): 273–292.

Leu, Y. Y., L. A. Matheson, and L. P. Rees. 1994. "Assembly-line Balancing Using Genetic Algorithms with Heuristic-generated Initial Populations and Multiple Evaluation Criteria." *Decision Sciences* 25 (4): 581–605.

Levitin, G., J. Rubinovitz, and B. Shnits. 2006. "A Genetic Algorithm for Robotic Assembly Line Balancing." *European Journal of Operational Research* 168 (3): 811–825.

Lusa, A. 2008. "A Survey of the Literature on the Multiple or Parallel Assembly Line Balancing Problem." *European Journal of Industrial Engineering* 2 (1): 50–72.

Ozbakir, Lale, Adil Baykasoglu, Beyza Gorkemli, and Latife Gorkemli. 2011. "Multiple-colony Ant Algorithm for Parallel Assembly Line Balancing Problem." *Applied Soft Computing* 11 (3): 3186–3198.

Ozbakir, L., and P. Tapkan. 2010. "Balancing Fuzzy Multi-objective Two-sided Assembly Lines via Bees Algorithm." *Journal of Intelligent & Fuzzy Systems* 21 (5): 317–329.

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. 2010. "Balancing Stochastic Two-sided Assembly Lines: A Chance-constrained, Piecewise-linear, Mixed Integer Program and a Simulated Annealing Algorithm." *European Journal of Operational Research* 205 (1): 81–97.

Ozcan, U., H. Cercioglu, H. Gokcen, and B. Toklu. 2009. "A Tabu Search Algorithm for the Parallel Assembly Line Balancing Problem." *Gazi University Journal of Science* 22 (4): 313–323.

Ozcan, U., H. Cercioglu, H. Gokcen, and B. Toklu. 2010. "Balancing and Sequencing of Parallel Mixed-model Assembly Lines." *International Journal of Production Research* 48 (17): 5089–5113.

Ozcan, U., H. Gokcen, and B. Toklu. 2010. "Balancing Parallel Two-sided Assembly Lines." *International Journal of Production Research* 48 (16): 4767–4784.

Ozcan, U., and B. Toklu. 2009. "A Tabu Search Algorithm for Two-sided Assembly Line Balancing." *The International Journal of Advanced Manufacturing Technology* 43 (7–8): 822–829.

Ozcan, U., and B. Toklu. 2010. "Balancing Two-sided Assembly Lines with Sequence-dependent Setup times." *International Journal of Production Research* 48 (18): 5363–5383.

Ozdemir, R. G., and Z. Ayag. 2011. "An Integrated Approach to Evaluating Assembly-line Design Alternatives with Equipment Selection." *Production Planning & Control* 22 (2): 194–206.

Purnomo, H. D., H. M. Wee, and H. Rau. 2013. "Two-sided Assembly Lines Balancing with Assignment Restrictions." *Mathematical and Computer Modelling* 57 (1–2): 189–199.

Rabbani, M., M. Moghaddam, and N. Manavizadeh. 2012. "Balancing of Mixed-model Two-sided Assembly Lines with Multiple U-shaped Layout." *The International Journal of Advanced Manufacturing Technology* 59 (9–12): 1191–1210.

Rekiek, B., P. De Lit, F. Pellichero, T. L'Eglise, P. Fouda, E. Falkenauer, and A. Delchambre. 2001. "A Multiple Objective Grouping Genetic Algorithm for Assembly Line Design." *Journal of Intelligent Manufacturing* 12 (5–6): 467–485.

Rubinovitz, J., and G. Levitin. 1995. "Genetic Algorithm for Assembly Line Balancing." *International Journal of Production Economics* 41 (1–3): 343–354.

Scholl, A., and N. Boysen. 2009. "Designing Parallel Assembly Lines with Split Workplaces: Model and Optimization Procedure." *International Journal of Production Economics* 119 (1): 90–100.

Simaria, A. S., and P. 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.

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.

Suresh, G., V. V. Vinod, and S. Sahu. 1996. "A Genetic Algorithm for Assembly Line Balancing." *Production Planning & Control* 7 (1): 38–46.

Suwannarongsri, S., S. Limnararat, and D. Puangdownreong. 2007. "A New Hybrid Intelligent Method for Assembly Line Balancing." *IEEE International Conference on Industrial Engineering and Engineering Management* 1–4: 1115–1119.

Taha, R. B., A. K. El-Kharbotly, Y. M. Sadek, and N. H. Afia. 2011. "A Genetic Algorithm for Solving Two-sided Assembly Line Balancing Problems." *Ain Shams Engineering Journal* 2 (3–4): 227–240.

Tonelli, F., M. Paolucci, D. Anghinolfi, and P. Taticchi. 2013. "Production Planning of Mixed-model Assembly Lines: A Heuristic Mixed Integer Programming Based Approach." *Production Planning & Control* 24 (1): 110–127.

Wee, T. S., and M. J. Magazine. 1982. "Assembly Line Balancing as Generalized Bin Packing." *Operations Research Letters* 1 (2): 56–58.

Wu, E. F., Y. Jin, J. S. Bao, and X. F. Hu. 2008. "A Branch-and-bound Algorithm for Two-sided Assembly Line Balancing." *The International Journal of Advanced Manufacturing Technology* 39 (9–10): 1009–1015.

Yegul, M. F., K. Agpak, and M. Yavuz. 2010. "A New Algorithm for U-shaped Two-sided Assembly Line Balancing." *Transactions of the Canadian Society for Mechanical Engineering* 34 (2): 225–241.

Yu, J. F., and Y. H. Yin. 2010. "Assembly Line Balancing Based on an Adaptive Genetic Algorithm." *The International Journal of Advanced Manufacturing Technology* 48 (1–4): 347–354.

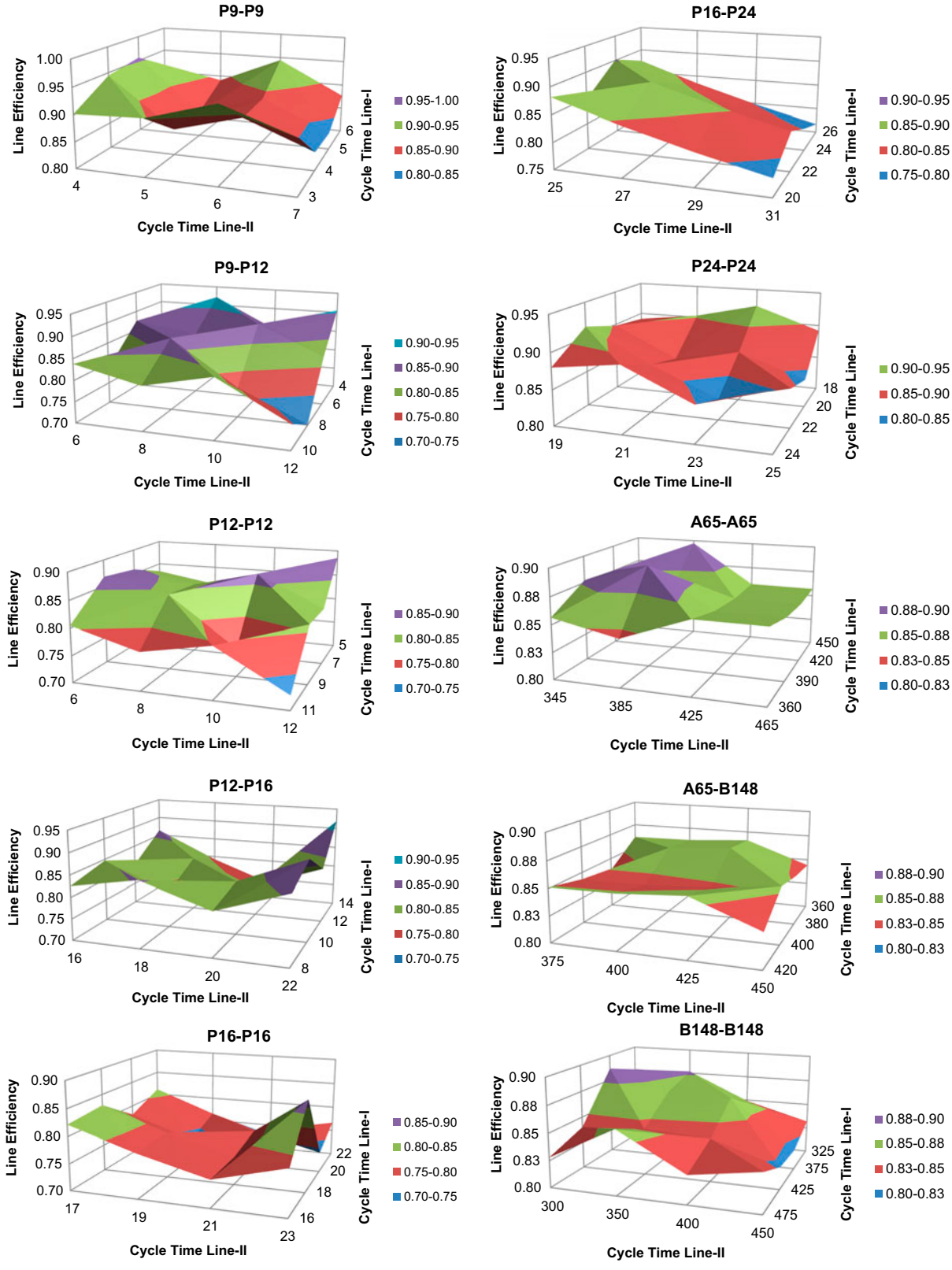
Zhang, W. Q., M. Gen, and L. Lin. 2008. "A Multiobjective Genetic Algorithm for Assembly Line Balancing Problem with Worker Allocation." *IEEE International Conference on Systems, Man and Cybernetics (SMC 2008)* 1–6: 3026–3033.

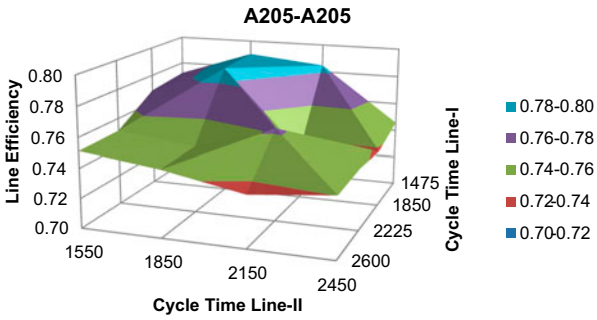
Zhang, Y. N., S. L. Kan, Y. Wang. 2005. "A Multi-objective Genetic-tabu Algorithm for the Assembly Line Balancing Problem." *Proceedings of the 11th International Conference on Industrial Engineering and Engineering Management* 1–2: 735–738.

Zhang, D. Z., and I. Kucukkoc. 2013. "Balancing Mixed-model Parallel Two-sided Assembly Lines." In *Proceedings of the International Conference on Industrial Engineering and Systems Management (IEEE-IESM'2013), École Mohammadia d'Ingénieurs de Rabat (EMI), International Institute for Innovation, Industrial Engineering and Entrepreneurship (I4E2)*, edited by L. Amodeo, A. Dolgui and F. Yalaoui, 391–401. Rabat: Morocco.

Appendix 1

A.1. Surface charts of the line efficiency values across the cycle time of Line-I and Line-II for the test problems.




A205-A205
Line Efficiency
0.80
0.78
0.76
0.74
0.72
0.70
1550
1850
2150
2450
Cycle Time Line-II
1475
1850
2225
2600
Cycle Time Line-I
0.78-0.80
0.76-0.78
0.74-0.76
0.72-0.74
0.70-0.72