Balancing of Parallel U-shaped Assembly Lines

Ibrahim Kucukkoc, David Z. Zhang

PII: S0305-0548(15)00141-0
DOI: http://dx.doi.org/10.1016/j.cor.2015.05.014
Reference: CAOR3796

www.elsevier.com/locate/caor

To appear in: *Computers & Operations Research*

Received date: 24 October 2014
Revised date: 8 April 2015
Accepted date: 30 May 2015

Cite this article as: Ibrahim Kucukkoc, David Z. Zhang, Balancing of Parallel U-shaped Assembly Lines, *Computers & Operations Research,* http://dx.doi.org/10.1016/j.cor.2015.05.014

Balancing of Parallel U-shaped Assembly Lines

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

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

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

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

Abstract

A new hybrid assembly line design, called Parallel U-shaped Assembly Line system, is introduced and characterised along with numerical examples for the first time. Different from existing studies on U-shaped lines, we combine the advantages of two individual line configurations (namely parallel lines and U-shaped lines) and create an opportunity for assigning tasks to multi-line workstations located in between two adjacent U-shaped lines with the aim of maximising resource utilisation. Utilisation of crossover workstations, in which tasks from opposite areas of a same U-shaped line can be performed, is also one of the main advantages of the U-shaped lines. As in traditional U-shaped line configurations, the newly proposed line configuration also supports the utilisation of crossover workstations. An efficient heuristic algorithm is developed to find well-balanced solutions for the proposed line configurations. Test cases derived from existing studies and modified in accordance with the proposed system in this study are solved using the proposed heuristic algorithm. The comparison of results obtained when the lines are balanced independently and when the lines are balanced together (in parallel to each other) clearly indicates that the parallelisation of U-shaped lines helps decrease the need for workforce significantly.

Keywords: assembly line balancing; parallel lines; U-shaped assembly lines; heuristic algorithm; production lines; artificial intelligence.

* Corresponding author: Ibrahim Kucukkoc, Permanent Email: ikucukkoc@balikesir.edu.tr
Tel: +441392723613 (I.K.); +441392723641 (D.Z.Z)

Acknowledgment

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

Balancing of Parallel U-shaped Assembly Lines

Abstract

A new hybrid assembly line design, called *Parallel U-shaped Assembly Line* system, is introduced and characterised along with numerical examples for the first time. Different from existing studies on U-shaped lines, we combine the advantages of two individual line configurations (namely parallel lines and U-shaped lines) and create an opportunity for assigning tasks to multi-line workstations located in between two adjacent U-shaped lines with the aim of maximising resource utilisation. Utilisation of crossover workstations, in which tasks from opposite areas of a same U-shaped line can be performed, is also one of the main advantages of the U-shaped lines. As in traditional U-shaped line configurations, the newly proposed line configuration also supports the utilisation of crossover workstations. An efficient heuristic algorithm is developed to find well-balanced solutions for the proposed line configurations. Test cases derived from existing studies and modified in accordance with the proposed system in this study are solved using the proposed heuristic algorithm. The comparison of results obtained when the lines are balanced independently and when the lines are balanced together (in parallel to each other) clearly indicates that the parallelisation of U-shaped lines helps decrease the need for workforce significantly.

Keywords: assembly line balancing; parallel lines; U-shaped assembly lines; heuristic algorithm; production lines; artificial intelligence.

1. Introduction

Flexibility is one of the most crucial criteria of modern manufacturing systems to satisfy customised demands in a cost-effective manner. As mentioned by Miltenburg [1], *proper design and machinery layout* is one of three important elements that help increase flexibility to respond to changes in demand. The other two elements are *multi-functional workers*, and *continuous evaluation and revisions of the standard operations*. U-shaped lines (or U-lines shortly) not only provide the flexibility and efficiency of a proper line design but also help increase functionality of workers on the line.

An assembly line is a sequence of workstations in which a set of tasks related to the assembly of a product are performed based on pre-defined rules and certain constraints. Precedence relationship constraint is one of these constraints and must be satisfied due to technological or organisational restrictions. Another must have constraint is capacity constraint. Tasks assigned to workstations must be completed within a designated time, called cycle time. Also, every task must be assigned to exactly one workstation to obtain a feasible balance. In U-shaped lines, the entrance and the exit of the line system are very close to each other and form a 'U' shape allowing operators to handle work-pieces both at the front and at the back of the line [2, 3]. Figure 1 provides schematic representation of a U-shaped line configuration.

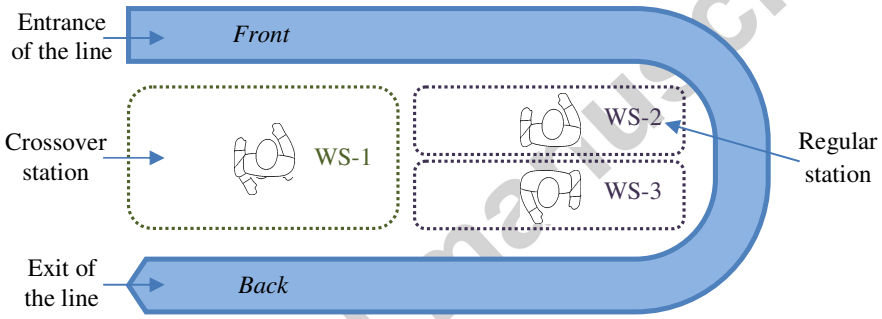


Figure 1. Typical illustration of a U-shaped line configuration

The common objective is to obtain the optimal line balance which minimises the total number of required workstations (or operators) for a given cycle time determined by demand for assembled product(s) on the line. This problem is referred to as type-I. The problem is called type-II if the minimisation of cycle time is the main objective for a given number of workstations [4]. Other two types of assembly line balancing problems in accordance with sought objective(s) are type-E and type-F. In type-E, both of the cycle time and total number of workstations are minimised while in type-F a feasible line balance is sought when both of the cycle time and number of workstations are given.

Many studies on new problem concepts including new line configurations (*e.g.* parallel, U-shaped, two sided, *etc.*), flexibility considerations (*e.g.* parallel workstations, mixed-models *etc.*), job-handling variations (*e.g.* manual or robotic lines, *etc.*), additional layout-based constraints (*e.g.* space

constraints, zoning constraints, *etc.*) and dynamism effects (*e.g.* changing model demands, learning effect, *etc.*) have been conducted for about six decades. However, hybrid configurations have become quite popular recently due to their advantages over individual configuration types because a hybrid line combines the advantages of both individual constituents. Moving from that point, we propose a hybrid line configuration, namely *parallel U-shaped assembly line system*, to increase resource utilisation and maximise line efficiency. Two U-shaped lines are located in parallel to each other to establish an assembly environment where operators located between two adjacent lines can handle tasks on both of the lines. The main contribution of this paper is that such a line system where a different model of a product is assembled on each of the U-shaped lines located in parallel is introduced for the first time in the literature. Each line may have a different cycle time from each other and the flexibility of the system increases even more by this way. Also, flexibility of the workstation utilisation in three different formats, *i.e. (i)* multi-line stations (between two adjacent lines), *(ii)* crossover stations (between front and back of the inner line) and *(iii)* regular stations (on only front or back of the inner line), help line managers move forward one step more towards successfully implemented JIT production systems. There is no doubt that the NP-Hard complexity of the problem makes its exact solution impossible using traditional techniques. For that reason, a new efficient heuristic algorithm is proposed to find well balanced solutions for parallel U-shaped assembly lines.

The remainder of the paper is organised as follows. In Section 2, a comprehensive review of the literature is presented. The parallel U-shaped assembly line balancing problem is characterised and illustrated in detail in Section 3. Section 4 and Section 5 present the proposed heuristic algorithm stepwise and two numerical examples, respectively. Section 6 reports the results of the performed computational tests followed by the conclusions and suggestions for future work in Section 7.

2. Review of the Literature

The first U-shaped assembly line design was brought to the attention of the academia by Miltenburg and Wijngaard [5]. They described the simplest form of a U-shaped assembly line balancing problem

and illustrated the advantages of utilising U-shaped assembly lines over traditional straight assembly lines. They developed *(i)* a dynamic programming approach which can solve problems with up to 11 tasks and *(ii)* a heuristic based technique for problems of larger size to minimise the number of workstations. This was followed by a series of publications. Table 1 presents a summary of the main contributions on U-shaped assembly lines in the literature.

Some exact (*e.g.* an integer programming formulation by Urban [6], a branch and bound procedure by Scholl and Klein [7]) and heuristic/meta-heuristic (*e.g.* a simulated annealing approach by Erel *et al.* [8], a shortest route formulation by Gökçen *et al*. [9], a genetic algorithm approach by Hwang *et al*. [10] and an ant colony optimisation based approach by Sabuncuoglu *et al*. [11]) solution approaches were developed to solve the U-shaped assembly line balancing problem in its traditional form where there is only one line on which a single model is produced. Among these studies, Scholl and Klein [7] and Hwang *et al*. [10] also aimed at minimising cycle time and obtaining a smooth workload distribution, respectively, as well as minimising the number of workstations as a common objective.

Large numbers of studies have been conducted on relatively more complex versions of U-shaped line balancing problems as well. Variability in processing times due to human factors (stochastic task times) on U-lines was studied by Chiang and Urban [12], Urban and Chiang [13] and Celik *et al*. [14]. Urban and Chiang [13] developed a chance-constrained piecewise-linear programme to optimally solve the U-shaped line balancing problem with stochastic task times. Hybrid heuristic and ant colony optimisation approaches were proposed by Chiang and Urban [12] and Celik *et al*. [14], respectively. While Chiang and Urban [12] aimed at minimising the number of workstations, Celik *et al*. [14] considered total cost of rebalancing as the primary goal. Learning effect in processing times is considered by Toksari *et al*. [15] when minimising the number of workstations. Kara *et al*. [16] and Jayaswal and Agarwal [2] proposed integer programming model and simulated annealing algorithm, respectively, to minimise operating costs.

Table 1. Summary of the main contributions on U-shaped assembly lines

Research	**Proposed Methodology**	**Main Obj.**			**Other Specifications**		
		K	**C**	**O**	**S**	**M**	**Explanation**
Miltenburg and Wijngaard [5]	A dynamic programming approach for small-sized problems and a	√			√		

Research	Proposed Methodology	Main Obj.			Other Specifications		
		K	C	O	S	M	Explanation
	heuristic technique for problems of larger size						
Urban [6]	Integer programming formulation	√			√		
Scholl and Klein [7]	Branch and bound procedure	√	√		√		Type-1, type-2, type-E
Erel *et al.* [8]	Simulated annealing approach	√			√		
Gökçen *et al*. [9]	Shortest route formulation	√			√		
Gökçen and Ağpak [17]	Goal programming approach	√	√	√	√		Number of tasks per workstations
Urban and Chiang [13]	Chance-constrained piecewise-linear programme	√			√		Stochastic task times
Chiang and Urban [12]	Hybrid heuristic algorithm	√			√		Stochastic task times
Kim *et al*. [18]	Endosymbiotic evolutionary algorithm			√		√	S-LB/MS, ADW
Kara *et al*. [19]	Simulated annealing algorithm based method			√		√	S-LB/MS, ADW, part usage rate, cost of setups
Chiang *et al.* [20]	Optimal solution methodologies	√	√		√		Cost-minimisation
Hwang *et al*. [10]	Multi objective genetic algorithm approach	√		√	√		Workload variation
Toksari *et al*. [15]	Heuristic approach	√			√		Learning effect
Kara *et al*. [21]	Binary fuzzy goal programming approach	√	√		√		Two or more conflicting objectives
Sabuncuoglu *et al*. [11]	An ant colony optimisation	√			√		
Yegul *et al*. [22]	Multi-pass random assignment algorithm	√			√		U-shaped two-sided assembly line
Kara *et al*. [16]	Integer programming formulation			√	√		Total operating cost, RDT
Özcan *et al*. [23]	Genetic algorithm approach			√		√	S-LB/MS, ADW with stochastic task times
Hamzadayi and Yildiz [24]	Genetic algorithm based approach	√		√		√	S-LB/MS, workload smoothness, parallel workstations and zoning constraints
Rabbani *et al.* [25]	Genetic algorithm approach	√	√			√	Two-sided line with multiple U-shaped layout
Hamzadayi and Yildiz [26]	Simulated annealing algorithm (enhanced with tabu list)	√				√	S-LB/MS
Manavizadeh *et al*. [27]	Simulated annealing algorithm	√				√	Human efficiency, workload variations, permanent and temporary workers
Battaïa and Dolgui [28]	Taxonomy of line balancing problems & solution approaches						Review of the assembly line balancing literature
Celik *et al*. [14]	Ant colony optimisation approach			√	√		Total cost of rebalancing, stochastic task times
Jayaswal and Agarwal [2]	Simulated annealing algorithm			√	√		Cost of workstation and resource utilization, RDT
Dong *et al*. [29]	Simulated annealing-based algorithm			√		√	S-LB/MS, expectation of work overload time, stochastic task times

K: number of stations, C: cycle time, O: other (see Explanation column), S: single model, M: mixed-model, RDT: resource dependent task times, S-LB/MS: simultaneous line balancing and model sequencing, ADW: absolute deviation of workloads

Also, to reflect real world conditions in industry, resource dependent task times for U-lines were considered in both studies. Manavizadeh *et al*. [27] considered human efficiency for mixed-model U-shaped assembly lines and set labour assignment policy after balancing the line using simulated annealing algorithm based on two types of operators: permanent and temporary. Yegul *et al.* [22] proposed multi-pass random assignment algorithm for U-shaped two-sided assembly line balancing problem with the aim of minimising the number of workstations. To help multi-criteria decision making process Gökçen and Ağpak [17] and Kara *et al*. [21] proposed goal programming and binary fuzzy goal programming approaches considering two or more conflicting objectives. In the former study, Gökçen and Ağpak [17] also considered the number of tasks assigned to each workstation as an additional goal.

Considering mixed-models on U-shaped lines requires extra care as it is possible that different model combinations can appear in the cross workstations depending on the model sequences produced on the line. Therefore, line balancing and model sequencing problems have been simultaneously considered by some of the researchers to find suitable balances for mixed-model U-lines. Kim *et al*. [18] and Hamzadayi and Yildiz [24] proposed evolutionary based algorithms while Kara *et al*. [19] and Hamzadayi and Yildiz [26] proposed simulated annealing algorithm based methods to solve the problem. Hamzadayi and Yildiz [24, 26] minimised number of workstations needed while Kim *et al*. [18] minimised absolute deviation of workloads (ADW) as primary goal. Hamzadayi and Yildiz [26] illustratively showed that ADW is an insufficient performance criterion for evaluating the performance of the solutions although it is a frequently used performance measure in the literature. Kara *et al*. [19] aimed at optimising part usage rate and cost of setups as well as ADW in a multi-objective manner. Özcan *et al*. [23] and Dong *et al*. [29] proposed genetic algorithm and simulated annealing based approaches, respectively, to solve the mixed-model assembly U-line balancing problem with stochastic task times. While ADW is considered as the performance measure by Özcan *et al*. [23], Dong *et al*. [29] aimed at minimising expectation of work overload time. Please refer to Özcan *et al*. [23] for a detailed review of the literature on balancing and sequencing of mixed-models

on U-shaped lines. Also Battaïa and Dolgui [28] presented a taxonomy of line balancing problems and their solution approaches.

Parallel line systems, a typical illustration of which is provided in Figure 2, have become quite popular recently (please see [30-33] for some real world examples). However, this concept – where two or more lines are located in parallel to each other – has not yet been studied for U-shaped lines, interestingly. Parallel line system was first proposed by Gökçen *et al*. [34] and have been hybridised with various types of other line configurations successfully; see for example Ozcan *et al*. [35], Kucukkoc and Zhang [36-38] and Kucukkoc *et al*. [39] for parallel two-sided lines; Özcan *et al*. [40] for parallel mixed-model lines; and Kucukkoc and Zhang [41-43] and Zhang and Kucukkoc [44] for mixed-model parallel two-sided lines. Gökçen *et al*. [34] showed that locating two straight lines in parallel to each other requires less number of workstations in comparison with performing the same tasks on individually balanced straight lines. Ozcan *et al*. [35] and Kucukkoc and Zhang [41] showed this advantage for the parallel two-sided lines and mixed-model parallel two-sided lines, respectively.

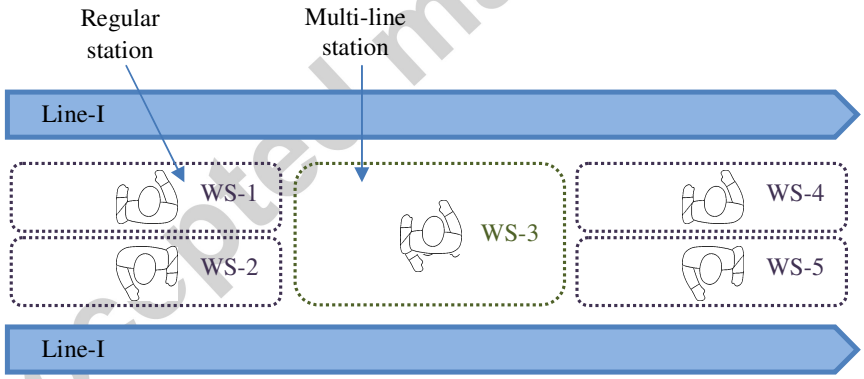


Figure 2. Schematic view of a parallel line system

Chiang *et al.* [20] introduced and formalised the multiple U-line balancing problem. They provided optimal solution methodologies for type-I, type-II and cost-minimisation assembly line balancing problems. Rabbani *et al.* [25] studied the mixed-model two-sided assembly line balancing problem with multiple U-shaped layout and proposed a genetic algorithm approach for the solution of the problem with the aim of minimising both the cycle time and the number of workstations. The most important advantage of the proposed parallel U-shaped assembly line configuration over these studies

is the flexibility of producing parts in different throughput rates. Also, the current work differs from these studies as stations may include tasks from multiple/independent lines, not adjacent lines as considered in the current work. This survey clearly contextualises the need for the current work to fill the gap in the literature.

3. Problem Description

U-shaped lines have emerged as a result of efforts on continuous improvement and cost reduction in just-in-time (JIT) production systems [21, 45]. The main reason lying behind this fact is that U-shaped lines provide more flexibility and possibility to assign tasks into workstations in many diversified groups or combinations. Thanks to crossover stations, tasks belonging to the front of the line can also be performed from the back of the line. Similarly, tasks belonging to the back of the line can be performed from the front of the line. This flexibility enables line managers to assign tasks into workstations in such a way that the utilisation of the line is maximised. It also helps obtain a smooth line balance where workload times are distributed among workstations as equally as possible.

This paper proposes an improved design of the traditional U-shaped lines where two or more U-lines, represented by L_h $(h = 1, \dots, H)$ are located in parallel to each other. Figure 3 represents a schematic view of the proposed parallel U-shaped assembly line system. As can be seen from the figure, working area is divided into *zones* $(z = 1,2, \dots ,4)$ and *queues* $(q = 1,2, \dots , Q;$ where Q is the total number of queues or length of the lines). Workstations (symbolised with W_q^z) located in between two adjacent lines are called *multi-line stations* (*e.g.* WS_1^1, WS_1^4, WS_2^1, *etc.*) and operators located in these workstations can perform their jobs on both of the lines. Workstations located in between front (beginning) and back (ending) of the inner line, which is Line-II in this representation, are called *crossover stations* (*e.g.* $WS_1^{2,3}$) and operators located in these stations can perform their jobs in either of the areas. There also are *regular stations* (*e.g.* WS_2^2 and WS_2^3), in which operators perform their jobs just for only one specific line and area (front or back).

One different product model is produced on each of the U-shaped assembly lines (m_h represents the model number produced on line L_h). Each product model produced on each of the lines has its own

set of tasks, where a task is represented by t_{hi} $(i = 1, \dots, T_h)$, performed according to predefined precedence relationships. There are certain precedence rules between the tasks and P_{hi} represents the set of predecessors of task t_{hi} for model m_h. Each task (t_{hi}) for model m_h on line L_h requires a certain amount of processing time (pt_{hi}) to be processed.

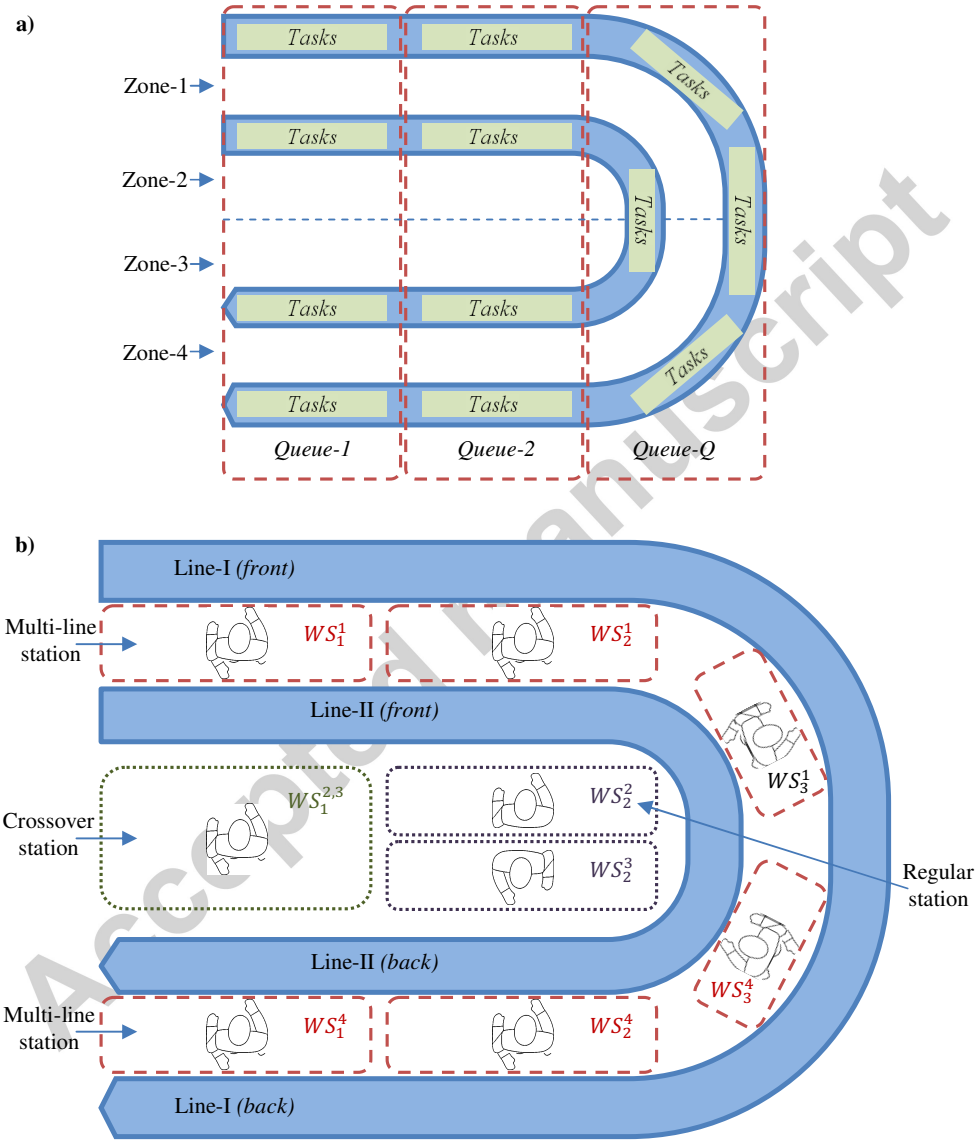


Figure 3. Schematic representation of the proposed parallel U-shaped assembly line system: a) zoning of the work area, b) allocation of the workstations

Cycle time of line L_h, represented by C_h, is calculated through dividing planning horizon (P) to model demand (D_h) produced on that particular line $(C_h = P/D_h)$. Each line may have different cycle time

and this is one of the major advantages of parallel U-shaped assembly lines, as each of the lines may have a different throughput rate in accordance with the demand of the model produced on that particular line. When parallel lines have different cycle times, a common cycle time should be used to allow operators work in multi-line stations with no conflict. Gökçen *et al.* [34] used the Least Common Multiple (*LCM*) approach, the main steps for which are given as follows:

- Find the *LCM* of the cycle times.
- Calculate the line divisors, ld_1 and ld_2 for Line-I and Line-II respectively, through dividing *LCM* by the cycle times of the Line-I and Line-II, respectively.
- Modify the processing times of tasks performed on Line-I and Line-II multiplying by ld_1 and ld_2 values, respectively.
- Consider the *LCM* as the common cycle time of all lines and balance the lines using modified processing times of tasks.

Having different throughput levels for two U-shaped lines provides flexibility of easy adaptation to changing model demands. Establishing two U-shaped lines in parallel to each other also provides flexibilities for assigning tasks in different groups or combinations and provides more opportunities to obtain even better line balances. The main advantages of the parallel U-shaped assembly lines, which combine the advantages of both parallel lines [34] and U-shaped lines [5] configurations, over independently balanced U-shaped lines are as follows:

- The flexibility in the throughput levels of the lines helps adapt changes in demands easily.
- The increased possibility of grouping tasks into different workstation types in various combinations yields more efficient balancing solutions where the number of required workstations is always less or equal to that required in traditional forms. Thus, resource utilisation is maximised thanks to the multi-line stations and crossover stations.
- Total number of required workers can be increased or decreased easily to adapt changes in demands.

- Work enrichment obtained through rotating operators between different working environments instead of repeating a single short cycle task increases the qualification of the operators and helps them stay more alert.
- Length of the line is shortened.
- Rotating operators between a wide range of working environments allows more workers to participate in efforts to improve the process.
- In case of any problem, *i.e.* break downs, workstations can get help and support from their companions.
- Multi-line and crossover workstations help increase communication skills between operators.

The problem of assigning tasks to different workstation types explained above in such a way that the total number of workstations is minimised is called *parallel U-shaped assembly line balancing problem*. Cycle times of the lines, which are determined based on the model demands, are given as input to the system. As in other types of assembly line balancing problems, certain constraints need to be satisfied while doing this. Each task must be assigned to exactly one workstation and the precedence relationship constraints must be assured. Also, sum of processing times of tasks assigned to a workstation (regardless of from one line or both of the lines) cannot exceed the capacity (cycle time) designated for this workstation. The assumptions considered in the study are as follows:

- Two U-shaped lines are located in parallel to each other.
- Operators are multi-skilled and can perform their jobs on either of the lines.
- Products (or product models) assembled on the lines carry one-sided assembly characteristics, *i.e.* there is no task which requires to be performed on a specific side. In our design, tasks on Line-I can be performed from only the inner side while tasks on Line-II can be performed from either of the sides.
- Demands of the models and processing times of the tasks are deterministic and known for a pre-determined planning horizon.

- Two different models are produced on the lines, *i.e.* one on each of the lines, and each model has its own set of tasks. Precedence diagrams for each product model are known.
- Operators can perform their jobs within designated area for themselves and walking times are ignorable. While operators located in multi-line stations (between two adjacent lines) can work on tasks for models assembled on both of the lines, operators in crossover stations can work on models assembled on front and back of the inner line (Line-II) only. The simplest version of the stations, regular station, is also allowed as can be seen from Figure 3.

The following subsection provides the heuristic approach proposed for balancing the parallel U-shaped assembly line system.

4. Proposed Solution Method

This section explains step-by-step the solution procedure of the Parallel U-line Heuristic (called *PUH* hereafter) developed to find well-balanced solutions for the parallel U-shaped assembly line balancing problem. To provide an overview, we first give the outline of the overall solution approach, and describe each of the steps in more detail.

The proposed algorithm employs modifications of a well-known heuristic, namely *Ranked Positional Weight Method* [46], and a variant of another well-known heuristic, namely *Maximum Number of Immediate Successors* [47], in specific to the characteristics of the parallel U-shaped assembly line balancing problem. The logic lying behind the assignment procedure is based on determining available tasks for a specific workstation, and selecting and assigning tasks to this workstation considering their priority indexes $\left(\omega_{hi}^{PI}\right)$. The pseudo-code of the solution method is given as follows:

Start by calculating task priority indexes (ω_{hi}^{PI}) for all tasks on Line-I and Line-II.

Initialise unassigned tasks lists (UTL_1 and UTL_2 for Line-I and Line-II, respectively) and zone-queue indexes ($z = 1$ and $q = 1$). To remind, W_q^z represents the workstation in zone z and queue q; where $z = 1, 2, \dots 4$ and $q = 1,2, \dots Q$.

Initialise all workstation times $\left(WT_q^z = 0;\ z = 1, 2, \dots 4;\ q = 1,2, \dots Q\right)$ and earliest starting times of all tasks $(EST_{hi} = 0; h = 1, 2;\ i = 1,2, \dots T_h)$.

Repeat **while** $UTL_1 \neq \emptyset$ or $UTL_2 \neq \emptyset$

Go to workstation W_q^z and determine all available tasks from Line-I and Line-II, where ATL_1 and ATL_2 are set of assignable tasks from Line-I and Line-II, respectively

Repeat **while** there are available tasks from Line-I or Line-II ($ATL_1 \neq \emptyset$ or $ATL_2 \neq \emptyset$)

Select and assign a task from the ATL_1 or ATL_2 in accordance with the task selection procedure.

Remove the assigned task from the relevant unassigned tasks list (UTL_1 or UTL_2).

Set the WT_q^z to the finishing time of the assigned task. Finishing time of a task is the maximum of the following two options: *i)* summation of the current workstation time $\left(WT_q^z\right)$ and the processing time of the assigned task (pt_{hi}), or *ii)* summation of the earliest starting time of the assigned task (EST_{hi}) and the processing time of the assigned task (pt_{hi}): $WT_q^z \leftarrow pt_{hi} + max\{WT_q^z, EST_{hi}\}$.

Update available tasks lists (ATL_1 and ATL_2) for the new situation.

If (the assigned task is from the front of the precedence relationships graph)

Update EST_{hi} values of its immediate successors

else if (the assigned task is from the back of the precedence relationships graph)

Update EST_{hi} values of its immediate predecessors

end if

End while

If $z \leq 3$, $z = z + 1$; otherwise $z = 1$ and $q = q + 1$.

End while

As can be seen from the given pseudo-code, the algorithm starts with calculating task priority indexes (ω_{hi}^{PI}). The ω_{hi}^{PI} value of a task is calculated using its positional weight (PW_{hi}) and total number of successors (NS_{hi}). The procedure followed when calculating the task priority indexes (ω_{hi}^{PI}) of tasks are as follows:

i. Calculate the positional weight of each task on the lines as follows:

$$PW_{hi} = pt_{hi} + \sum_{j \in S_{hi}} pt_{hj}, \quad (1)$$

where S_{hi} represents the set of successors of task t_{hi} on line L_h.

ii. On each of the lines, sequence all tasks in ascending order based on their PW_{hi} values, and assign '1' to the positional weight index (ω_{hi}^{PW}) of task which has the lowest PW_{hi} value. Select the next task which has the second lowest PW_{hi} value and assign it positional weight index of 2; and so on until all tasks are assigned a positional weight index. For example, if tasks 9 and 7

on Line-I have the lowest and the second lowest positional weights, respectively, then $\omega_{19}^{PW} = 1$ and $\omega_{17}^{PW} = 2$. If there are tasks which have the same positional weights, they are assigned the same positional weight indexes. By this way, positional weight index of a task can be the total number of tasks on that line (T_h) at maximum. This is only possible when there are no tasks with the same positional weights at all.

iii. Calculate the total number of successors of each task on the lines as follows:

$$NS_{hi} = card\{S_{hi}\}, \tag{2}$$

where $card\{S_{hi}\}$ corresponds to the length of the set of successors of task t_{hi} on line L_h.

iv. On each of the lines, sequence all tasks in ascending order based on their NS_{hi} values, and assign '1' to the number of successors index (ω_{hi}^{NS}) of task which has the lowest NS_{hi} value. Select the next task which has the second lowest NS_{hi} value and assign its ω_{hi}^{NS} value 2; and so on. For example, if tasks 8 and 9 on Line-II have the lowest and the second lowest total number of successors, then $\omega_{28}^{NS} = 1$ and $\omega_{29}^{NS} = 2$. Again, ω_{hi}^{NS} values of tasks with the same NS_{hi} values will be the same.

v. Calculate the overall priority index for every task as follows:

$$\omega_{hi}^{PI} = \omega_{hi}^{PW} \times \omega_{hi}^{NS}. \tag{3}$$

After calculating task priority indexes, unassigned tasks lists and zone-queue indexes, which help to identify workstations, are initialised. Workload times of all workstations and earliest starting times of all tasks are set to zero. The assignment procedure starts from the workstation located in the first zone and first queue, W_1^1. All available tasks from both of the lines are determined for the current workstation and available tasks lists (ATL_1 and ATL_2) are constructed. The criteria checked to determine whether a task is available are as follows:

- The task has not been assigned.
- The task has no preceding (succeeding) tasks or all of its predecessors (successors) have already been assigned and completed if the task is assigned from the front (back) of the precedence relationships graph.

- Remaining capacity in the current workstation is large enough to perform the task.

To check whether there is enough capacity, earliest starting times of all tasks are recorded because some of the predecessors (or successors) of a task can be completed from another workstation. Keeping this information and checking it during the assignment procedure are important to avoid infeasible balancing solutions. Therefore, as noted in the pseudo-code of the algorithm, this constraint is satisfied if the following expression is fulfilled: $C - WT_q^z \geq pt_{hi} + max\{WT_q^z, EST_{hi}\}$.

The EST_{hi} value of task t_{hi} is determined by its predecessors (successors) if the task is assigned from the front (back) of the precedence relationships graph. To clarify, assigning a task from the front (back) of the precedence relationships graph means that the task assigned has no predecessors (successors) and is determined as 'available' and included in the available tasks list (ATL_1 or ATL_2) based on this factor. For example, let us assume $t_{ha}, t_{hb} \in P_{hi}$ and tasks t_{ha} and t_{hb} from the front of the precedence relationships graph are assigned to the current queue and are completed in the ninth and seventh time points in their workstations' cycle time, respectively. Then task t_{hi} can start in the current queue when both of the tasks are completed ($EST_{hi} = 9$). However, if t_{ha} has been assigned to one previous queue than t_{hb} and t_{hi}; then EST_{hi} value of task t_{hi} would be 7, not 9 ($EST_{hi} = 7$) as task t_{ha} would have already been completed in the previous queue. This situation would be subject to consideration for the predecessors of tasks t_{ha} and t_{hb} if they were being assigned from the back of the line. Please see the example given in Section 5.

When the lists of available tasks are constructed, a task is selected and assigned to the current position (W_q^z) and the workstation time (WT_q^z) is increased by the amount of assigned task's processing time. Among the lists of available tasks from both of the lines, tasks are selected and assigned to the workstations in accordance with the designated task selection strategy. While selecting tasks from the available tasks lists, randomness is allowed to scan the search space more effectively and obtain more diversified solutions. Randomness is a commonly applied technique in assembly line balancing approaches when selecting tasks and allocating to workstations, especially in priority rule-based methods. As mentioned by Otto and Otto [48], the quality of the solutions obtained by a priority rule-

based method can be improved by 'applying several passes of this priority rule with some kind of random influence'. Therefore, a random number, $r_1 \in (0,1)$, is determined by the algorithm. If this number is smaller than or equal to the randomness index, $RI \in [0,1]$, determined by the user at the beginning and given as an input to the algorithm ($r_1 \leq RI$), a random task is selected among the available ones. If the random number determined by the algorithm is greater than the randomness index ($r_1 > RI$), then the task which has the best priority index $\left(\omega_{hi}^{PI}\right)$ among the available tasks is selected. Best priority index depends on the position of the available task. In other words, if the available task is from the back of the precedence relationships graph (regardless of Line-I or Line-II), the best priority index corresponds to minimum ω_{hi}^{PI}. However, if the available task is from the front of the precedence relationships graph (regardless of Line-I or Line-II), the best priority index corresponds to the maximum ω_{hi}^{PI} value.

Another randomly determined number, $r_2 \in (0,1)$, is used to decide on selecting task from which of the lists, *i.e.* ATL_1 or ATL_2. If the randomly determined number is equal to or lower than 0.5, *i.e.* $r_2 \leq 0.5$, the algorithm tries to assign a task from ATL_1. If ($r_2 > 0.5$), the algorithm tries to assign a task from ATL_2. If the list the algorithm tries to assign a task from which is empty, then the task is selected from the other list. The task selection rule employed in this study can be expressed as follows:

$$t_{hi} = \begin{cases} \textit{random task selection}, & \textit{if } r_1 \leq RI \\ \arg\max_{t_{hi} \in ATL_1} \{\omega_{hi}^{PI}\}, & \textit{if } r_1 > RI \ \wedge \ r_2 \leq 0.5 \ \wedge \textit{assigning a task from front of PRG} \\ \arg\min_{t_{hi} \in ATL_1} \{\omega_{hi}^{PI}\}, & \textit{if } r_1 > RI \ \wedge \ r_2 \leq 0.5 \ \wedge \textit{assigning a task from back of PRG} \\ \arg\max_{t_{hi} \in ATL_2} \{\omega_{hi}^{PI}\}, & \textit{if } r_1 > RI \ \wedge \ r_2 > 0.5 \ \wedge \textit{assigning a task from front of PRG} \\ \arg\min_{t_{hi} \in ATL_2} \{\omega_{hi}^{PI}\}, & \textit{if } r_1 > RI \ \wedge \ r_2 > 0.5 \ \wedge \textit{assigning a task from back of PRG} \end{cases} \quad (4)$$

where PRG denotes the precedence relationships graph. The lists of available tasks (ATL_1 and ATL_2) are updated every time when a new task is assigned. Afterwards, another new task is selected and assigned to the current workstation and this cycle continues until there is no task in either of the ATL_1 or ATL_2 lists. When both of these lists are empty, zone index is increased by one ($z = z + 1$) if $z \leq 3$. In case of $z > 3$, queue index is increased by one ($q = q + 1$) and zone index is set to one ($z =$

1). Again ATL_1 and ATL_2 are updated and new tasks are assigned to the current position and this cycle continues until all tasks belonging to Line-I and Line-II are assigned to exactly one workstation.

The following section provides practical examples on how the proposed heuristic algorithm works to find balancing solutions for the proposed parallel U-shaped assembly line configuration.

5. Numerical Examples

5.1. Example 1

Let us consider two different models of a product, Model-I and Model-II, assembled on a parallel U-shaped assembly line system, *i.e.* Model-I on Line-I and Model-II on Line-II. The common precedence diagram of the models is given in Figure 4 while processing times of tasks are provided in Table 2 for both models. The cycle time is considered as 10 time units for both of the lines.

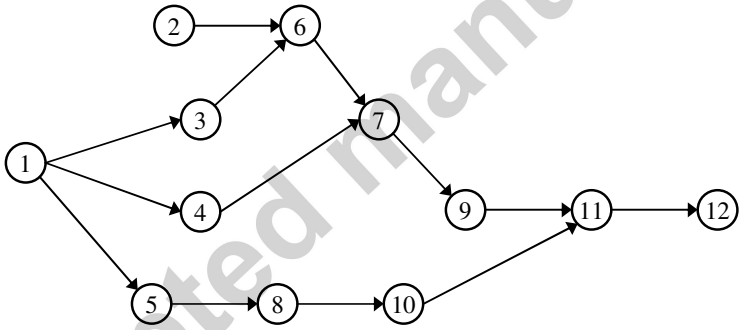


Figure 4. Common precedence diagram of the models for the numerical example

Table 2. Processing times of the tasks for the numerical example

Task Number	Model-I (in time units)	Model-II (in time units)
1	5	5
2	2	2
3	2	1
4	5	5
5	2	2
6	3	3
7	4	7
8	7	4
9	4	2
10	4	4
11	2	3
12	3	3
-	$\sum_{i=1}^{T_1} pt_{1i} = 43$	$\sum_{i=1}^{T_2} pt_{2i} = 41$

Table 3. Calculated priority indexes of tasks

Task Number	Model-I					Model-II				
	PW_{hi}	NS_{hi}	ω_{hi}^{PW}	ω_{hi}^{NS}	ω_{hi}^{PI}	PW_{hi}	NS_{hi}	ω_{hi}^{PW}	ω_{hi}^{NS}	ω_{hi}^{PI}
1	41	10	7	7	49	39	10	11	7	77
2	18	5	6	6	36	20	5	10	6	60
3	18	5	6	6	36	19	5	9	6	54
4	18	4	6	5	30	20	4	10	5	50
5	18	4	6	5	30	16	4	7	5	35
6	16	4	5	5	25	18	4	8	5	40
7	13	3	4	4	16	15	3	6	4	24
8	16	3	5	4	20	14	3	5	4	20
9	9	2	3	3	9	8	2	3	3	9
10	9	2	3	3	9	10	2	4	3	12
11	5	1	2	2	4	6	1	2	2	4
12	3	0	1	1	1	3	0	1	1	1

Table 4. Steps of the assignment procedure for the numerical example

Step	Workstation	ATL_1	ATL_2	Selected Task	WT_q^z	C-WT_q^z
1	W_1^1	1	1	$1 \in ATL_1$	5	5
2	W_1^1	2,3,4,5	1	$3 \in ATL_1$	7	3
3	W_1^1	2,5	1	$2 \in ATL_1$	9	**1**
4	$W_1^{2,3}$	4,5,6	1,12	$1 \in ATL_2$	5	5
5	$W_1^{2,3}$	4,5,6	2,3,4,5,12	$5 \in ATL_2$	7	3
6	$W_1^{2,3}$	5,6	2,3,12	$12 \in ATL_2$	10	**0**
7	$W_1^{3,4}$	12	11	$12 \in ATL_1$	3	7
8	$W_1^{3,4}$	11	11	$11 \in ATL_1$	5	5
9	$W_1^{3,4}$	9,10	11	$11 \in ATL_2$	8	**2**
10	$W_2^{1,2}$	4,5,6	2,3,4,8	$2 \in ATL_2$	2	8
11	$W_2^{1,2}$	4,5,6	3,4,8	$4 \in ATL_1$	7	3
…	…	…	…	…	…	…
22	$W_1^{3,4}$	7	8,9	$9 \in ATL_2$	2	8
23	$W_1^{3,4}$	7	8	$8 \in ATL_2$	6	4
24	$W_1^{3,4}$	7	-	$7 \in ATL_1$	10	**0**

To start the algorithm, task priority indexes need to be calculated first. The calculated priority indexes of tasks for the given numerical example are given in Table 3. The randomness index (RI) is assumed $RI = 0.5$ to have a large variety of balancing solutions.

The steps of the sample assignment procedure using the task selection rule given in the previous section are presented in Table 4. As can be seen from the table, the workload and the remaining capacity of the relevant workstation are also given after each new assignment. The algorithm starts assigning tasks from the first workstation after determining the available tasks for this position and continues by picking up and assigning one available task from the list of available tasks updated after

every task assignment. This cycle continues until there is no enough capacity in the current workstation. Afterwards, the algorithm moves to the next workstation and the assignment procedure continues by this way until all tasks belonging to both of the lines are assigned to a workstation.

The final assignment configuration of tasks on the proposed line system is given in Figure 5. As can be distinguished, a total of 9 nine workstations (3 of which are multi-line stations and 2 of which are crossover stations) are utilised across the lines. Table 5 presents final assignment of tasks, and workloads - idle times of the workstations.

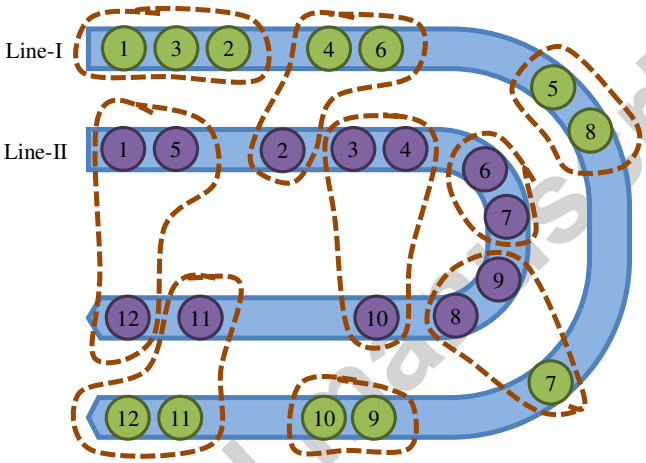


Figure 5. Allocation of the tasks in the final balancing solution

Table 5. Task assignments and idle times of the workstations

Workstation	Assigned Tasks		WT_q^z	$C - WT_q^z$
	Line-I	Line-II		
W_1^1	1,3,2	-	9	1
$W_1^{2,3}$	-	1,5,12	10	0
$W_1^{3,4}$	12,11	11	8	2
$W_2^{1,2}$	4,6	2	10	0
$W_2^{2,3}$	-	3,4,10	10	0
W_2^4	10,9	-	8	2
W_3^1	5,8	-	9	1
W_3^2	-	6,7	10	0
$W_3^{3,4}$	7	9,8	10	0

If these U-lines are balanced independently, not in parallel to each other, the theoretical minimum numbers of workstations ($TMNW_1$ and $TMNW_2$) for independent balances of Line-I and Line-II, respectively, can be calculated as $TMNW_1 = \lceil \sum_{i=1}^{T_1} pt_{1i}/C \rceil = \lceil 43/10 \rceil = 5$ and $TMNW_2 =$

$\left\lceil \sum_{i=1}^{T_2} pt_{2i}/C \right\rceil = \lceil 41/10 \rceil = 5$, where the expression $\lceil X \rceil$ corresponds to the smallest integer that is greater than or equal to X. In this scenario, a total of 10 workstations are needed to perform 24 tasks on both of the individual lines. That means one workstation more than the parallel U-shaped line configuration. Therefore, locating these U-lines in parallel to each other helps reduce the number of workstations at first glance. Other advantages of parallel U-shaped assembly lines have already been emphasised in Section 3. In addition, line efficiency values of Line-I and Line-II (LE_1 and LE_2, respectively) can be calculated as $LE_1 = \left(\sum_{i=1}^{T_1} pt_{1i}\right)/(C \times TMNW_1) \times 100 = 86\%$ and $LE_2 = \left(\sum_{i=1}^{T_2} pt_{2i}\right)/(C \times TMNW_2) \times 100 = 82\%$ in the individual balancing scenario. However, the line efficiency of the overall system is calculated as 93.3% when these two U-shaped lines are located in parallel to each other and are balanced together. Thus, an improvement of 11% is achieved in the efficiency of the overall line system over the arithmetic average of LE_1 and LE_2. Please note that the comparison with the arithmetic average of LE_1 and LE_2 is valid only when the lines have the same cycle times. Please refer to Kucukkoc and Zhang [37] to find out more about calculating overall system efficiency of a parallel line system when the parallel lines have different cycle times.

5.2. Example 2

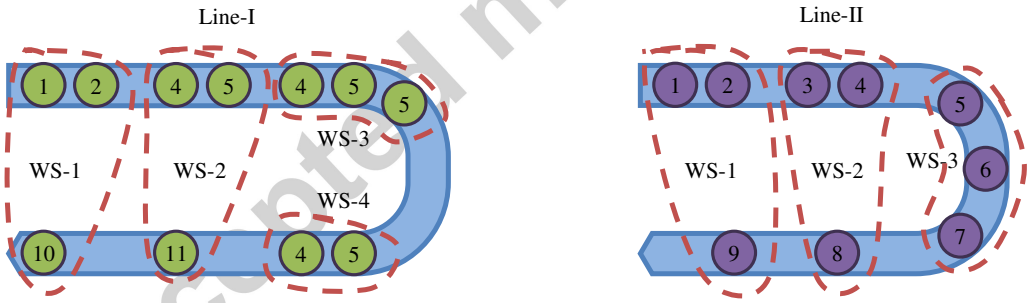
Another simple example is given here to exhibit the advantage of balancing two U-shaped assembly lines together over balancing two U-shaped assembly lines independently. Different from the numerical example given in Section 5.1; two different well-known test problems are considered on the lines, namely *Jackson (11-tasks)* on Line-I and *Jaeschke (9-tasks)* on Line-II, and the lines are balanced with different cycle times (please note that the test problems are taken from the type-I U-shaped assembly line balancing problem data set available at [49]). Table 6 provides the tasks processing times and precedence relationships data for the problems considered.

The cycle times are assumed 14 time units and 18 time units for Line-I and Line-II, respectively. Since different cycle times are considered for the lines, a common cycle time needs to be constructed for the utilisation of multi-line stations, as mentioned in Section 3. As explained, the *LCM* of the cycle times, and the line divisors are calculated ($LCM(14,18) = 126$; $ld_1 = 126/14 = 9$; and $ld_2 =$

$126/18 = 7$); and the task times are normalised, *i.e.* processing time of each task on Line-I is multiplied by ld_1 and processing time of each task on Line-II is multiplied by ld_2. The *LCM* value is considered as the common cycle time for both of the lines and the lines are balanced using the normalised task times.

Table 6. Precedence relationships and task processing times for the numerical example

Line-I ***(Jackson, 11-tasks)***			**Line-II** ***(Jaeschke, 9-tasks)***		
Task No	Task Processing Time	Immediate Successors	Task No	Task Processing Time	Immediate Successors
1	6	2,3,4,5	1	5	2,3
2	2	6	2	3	4
3	5	7	3	4	4
4	7	7	4	5	5,6,7
5	1	7	5	4	8
6	2	8	6	5	9
7	3	9	7	1	9
8	6	10	8	4	9
9	5	11	9	6	-
10	5	11	-	-	-
11	4	-	-	-	-
$\sum_{i=1}^{T_1} pt_{1i} = 46$			$\sum_{i=1}^{T_2} pt_{2i} = 37$		



Station Workloads

Line-I	WS-1	117	Line-II	WS-1	98
	WS-2	108		WS-2	91
	WS-3	90		WS-3	70
	WS-4	99		-	-

Figure 6. Task assignments when the lines are balanced independently

Figure 6 depicts possible line balancing solutions for the optimal number of workstations when the lines are balanced independently. We know that the solutions are optimal because the obtained station numbers for each problem are equal to their theoretical lower bounds (LB_K) for total number of

required workstations (LB_K can be calculated for Line-I and Line-II as $\lceil \sum_{i=1}^{T_1} pt_{1i} / C_1 \rceil = \lceil 3.29 \rceil = 4$ and $\lceil \sum_{i=1}^{T_2} pt_{2i} / C_2 \rceil = \lceil 2.06 \rceil = 3$, respectively).

If the lines are balanced together as proposed in this study, a line configuration given in Figure 7 can be obtained this time. As distinguished from the figure, a total of 6 workstations are needed when the lines are balanced together by allowing the utilisation of multi-line stations between two adjacent lines. This proves that balancing U-shaped lines together help save number of workstations (or operators) in comparison with independent balancing which requires a total of 7 workstations.

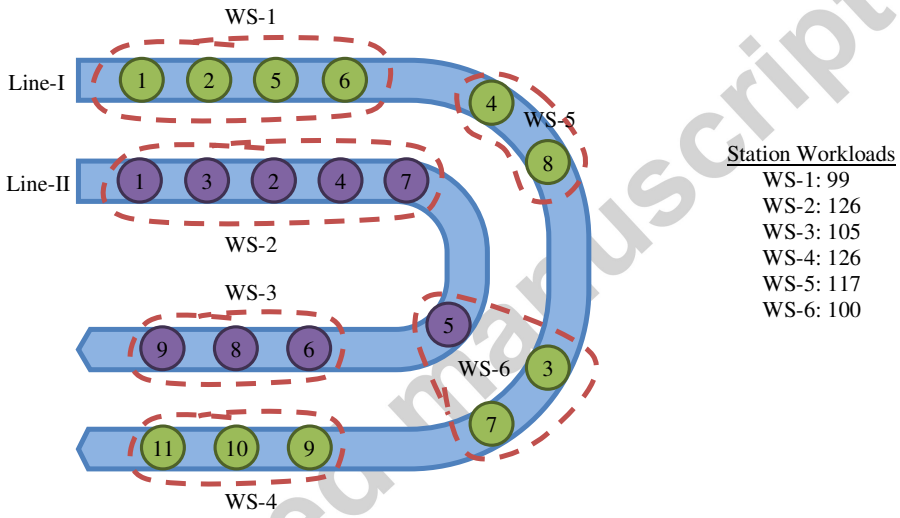


Figure 7. Task assignments when the lines are balanced together

Calculation of LB_K for each independent line balance was provided earlier. However, following equation can be used to calculate the LB_K for total number of required workstations across the lines when the lines are balanced together:

$$LB_K = \left\lceil \frac{\sum_{h=1}^{H} \left(\sum_{i=1}^{T_h} pt_{hi} \times C / C_h \right)}{C} \right\rceil = \left\lceil \sum_{h=1}^{H} \left(\sum_{i=1}^{T_h} pt_{hi} / C_h \right) \right\rceil \quad (5)$$

where C denotes the designated common cycle time ($C = LCM(C_1, C_2)$) and $\lceil X \rceil$ corresponds to the smallest integer that is greater than or equal to X.

Regarding the example given here, LB_K can be calculated simply as $LB_K = \lceil 5.35 \rceil = 6$ which means that the solution found when balancing the lines together is optimal. To remember, LB_K was

calculated as $\lceil 3.29 \rceil + \lceil 2.06 \rceil = 7$ when we balance the lines independently but it equals to $\lceil 5.35 \rceil = 6$ when we balance the lines together. It is apparent that balancing the lines together creates more opportunities for assigning tasks in various combinations and reduces LB_K.

6. Experimental Tests

A set of experimental tests are conducted to observe the effect of balancing lines together on total number of required workstations. The algorithm is coded in JAVA™ SE 7u4 environment and run on a 3.1 GHz Intel Core™ i5-2400 CPU computer. Test problems solved by Balakrishnan *et al.* [50] (and available at [49]) are taken as input data. In Balakrishnan *et al.* [50], one test problem is solved in each test case due to the nature of the assembly line structure. However, as we consider two U-shaped lines in parallel to each other, each test problem solved by Balakrishnan *et al.* [50] is accommodated on each of the parallel U-shaped assembly lines in our cases. Therefore, in this work, a pair of test problems -one on each of the lines- is solved at the same time for each test case.

Table 7 gives the experimental design used for the computational tests and reports the comparison of the obtained results in this study against the quite competitive independent balancing solution results available in the literature. In the table, P and C columns present the test problem and cycle time considered on each particular line. For example, two different test problems, namely *Jaeschke (9-tasks)* and *Mertens (7-tasks)*, are considered on Line-I and Line-II, respectively, for test case #7 with the original cycle times of the lines (*i.e.* $C_1 = 6$ and $C_2 = 8$). *OPT* column provides the optimal solutions of the test problems considered when two U-shaped lines are balanced independently where utilisation of multi-line stations between the lines is not allowed while utilisation of crossover workstations is allowed. Similarly, for each test problem, *ULINO*, *CSBM*, *SABM* and *MAXRP* report the independent balancing solutions available in the literature. Calculation of lower bound for total number of workstations when two parallel U-shaped assembly lines are balanced together was given in Equation 5 in Section 5. LB_K column gives this value for each solved test case.

Table 7. Experimental design and computational results of the experimental study

Test Case #	Line-I		Line-II		Balancing Independently (K value for L1+L2)		Balancing Together (PUH)				
					OPT^a, $ULINO^a$, $CSBM^b$, $SABM^c$	$MAXRP^d$	LB_K	$RI = 0.0$		$RI = 0.5$	
	P	C	P	C				LL	K	LL	K
1	Mertens 7-Tasks	6	Mertens 7-Tasks	6	12	12	10	3	11	3	***10***
2		6		7	11	11	9	3	11	3	**10**
3		7		7	10	10	9	3	10	3	***9***
4		8		8	10	10	8	3	**9**	3	**9**
5		8		15	7	7	6	2	7	2	***6***
6	Jaeschke 9-Tasks	6	Mertens 7-Tasks	6	14	14	11	4	13	3	**12**
7		6		8	13	13	10	4	**12**	4	**12**
8		7		8	12	12	9	4	12	3	**11**
9		8		8	11	11	9	3	**10**	3	**10**
10		18		15	5	5	4	2	5	1	***4***
11	Jaeschke 9-Tasks	6	Jaeschke 9-Tasks	7	15	15	12	4	**15**	4	**14**
12		6		8	14	14	11	4	**13**	4	**13**
13		8		18	9	9	7	3	9	4	**8**
14	Jackson 11-Tasks	9	Jaeschke 9-Tasks	6	14	14	12	4	14	4	**13**
15		10		6	13	13	11	4	13	4	**12**
16		14		8	10	10	8	3	**9**	3	**9**
17		14		18	7	7	6	2	***6***	2	***6***
18		21		18	6	6	5	2	***5***	2	***5***
19	Jackson 11-Tasks	9	Jackson 11-Tasks	10	11	11	10	3	11	3	***10***
20		10		14	9	9	8	3	9	3	9
21		14		21	7	7	6	2	***6***	2	***6***
22	Mitchell 21-Tasks	14	Jackson 11-Tasks	9	14	14	13	5	14	5	14
23		15		14	12	12	11	5	12	5	12
24		21		10	10	11	10	3	11	3	*10*
25		21		21	8	9	8	4	9	3	*8*
26	Mitchell 21-Tasks	14	Mitchell 21-Tasks	15	16	16	15	4	16	4	16
27		15		15	16	16	14	4	**15**	4	**15**
28		15		21	13	14	12	4	14	4	13
29	Heskiaoff 28-Tasks	138	Mitchell 21-Tasks	15	16	16	15	4	***15***	4	***15***
30		205		14	13	14	13	4	*13*	4	*13*
31		205		21	10	12	10	3	11	3	*10*
32		216		21	10	11	10	3	*10*	3	*10*
33	Heskiaoff 28-Tasks	138	Heskiaoff 28-Tasks	205	13	14	13	4	*13*	4	*13*
34		205		216	10	11	10	3	*10*	3	*10*
35		216		324	9	9	8	2	***8***	2	***8***
36	Kilbridge 45-Tasks	79	Heskiaoff 28-Tasks	138	15	16	15	4	16	4	*15*
37		110		205	11	12	11	3	*11*	3	*11*
38		110		216	11	11	10	3	***10***	3	***10***
39	Kilbridge 45-Tasks	57	Kilbridge 45-Tasks	79	17	18	17	5	18	5	*17*
40		92		110	12	13	12	3	*12*	3	*12*
41		110		110	12	12	11	3	***11***	3	***11***
42	Tonge 70-Tasks	364	Kilbridge 45-Tasks	79	17	18	17	5	18	6	*17*
43		410		79	16	17	16	5	17	4	*16*
44		468		110	14	14	13	4	14	4	***13***
45		527		110	13	13	12	4	13	3	***12***

Table 7 (continued).

Test Case #	Line-I		Line-II		Balancing Independently (K value for L1+L2)		Balancing Together (PUH)				
					OPT^a, $ULINO^a$, $CSBM^b$, $SABM^c$	$MAXRP^d$	LB_K	$RI = 0.0$		$RI = 0.5$	
	P	C	P	C				LL	K	LL	K
46	Tonge 70-Tasks	364	Tonge 70-Tasks	410	19	19	19	5	*19*	5	*19*
47		410		468	17	17	17	5	*17*	5	*17*
48		468		527	15	15	15	4	*15*	4	*15*
49	Arcus 83-Tasks	5048	Tonge 70-Tasks	364	26	26	25	8	26	8	26
50		6842		468	20	20	19	6	20	6	20
51		8898		527	16	16	16	5	*16*	4	*16*
52	Arcus 83-Tasks	6842	Arcus 83-Tasks	7571	23	23	22	6	23	6	23
53		7571		7571	22	22	20	6	**21**	6	**21**
54		7571		8412	21	21	19	5	**20**	5	**20**
55		8412		6842	22	22	21	6	***21***	6	***21***
56		8898		8412	19	19	18	5	19	5	19

** Please note that the results obtained from ULINO[a], CSBM[b] and SABM[c] are presented in the same column with OPT[a], as they give the same result. a: Scholl and Klein [7], b: Ağpak et al. [51], c: Baykasoğlu [52], d: Balakrishnan et al. [50].*

The results obtained using the parallel U-shaped line system and the heuristic proposed in this study are exhibited in *PUH* column for $RI = 0.0$ and $RI = 0.5$. When $RI = 0.0$, there is no randomness index considered in the task selection process and the tasks are selected purely based on their priority index values as explained in Section 4, so that, the effect of considering RI on quality of the obtained solutions can be examined. *PUH* algorithm is run for 100, 150 and 200 iterations for test cases #1 – #20, #21 – #40 and #41 – #60, respectively, in both levels of RI (*i.e.* $RI = 0.0$ and $RI = 0.5$). For each test case considered, the solution giving the minimum number of workstations is favoured and taken as the best solution when the algorithm is terminated. Line length (total number of queues utilised) and total number of workstations of the obtained best solutions are given in *LL* and *K* columns, respectively. Table 7 shows the advantages of using the proposed parallel U-shaped assembly line configuration. To do so, the results obtained by *PUH* algorithm when $RI = 0.5$ should be compared with the optimal results of independent balancing solutions presented in *OPT* column. The proposed line system helps save two workstations in two test cases, *i.e.* test cases #1 and #6; and one workstation for 28 test cases, *i.e.* test cases #2 – #5, #7 – #19, #21,

#27, #29, #35, #38, #41, #44, #45, and #53 – #55 (given in bold font in the table). It should be noted here that it is not possible to have a together balancing solution better than the independent balancing solution for 16 test cases. This situation is valid if the LB_K value of a test case is equal to the optimal result of independent balancing solution given in *OPT* column for this test case (*i.e.* see test cases #24, #30, #31, *etc.*).

The computational results presented here provide a measure in terms of the quality of the solutions found by the proposed algorithm as well. As it is not possible to obtain a solution which gives less number of workstations in comparison to the LB_K value calculated for a test case, we can say that the solution found is optimal if the obtained *K* value for this test case is equal to its LB_K value (see test cases #1, #3, #5, #10, #17 – #19, #21, #24, #25, #29 – #48, #51, and #55 when $RI = 0.5$). When the results presented in *K* ($RI = 0.5$) and LB_K columns are compared, it is clear that the algorithm proposed in this research finds optimal solutions for a minimum of 32 test cases out of 56 test cases. To provide a better understanding, these results are provided in italic font in the table.

The results presented in the table also give an idea about the effect of considering RI parameter for the proposed *PUH* procedure developed in this research. Considering RI parameter as $RI = 0.5$ helps algorithm find better solutions in terms of *K* values for 22 test cases in comparison to results obtained when $RI = 0.0$ (*e.g.* see test cases 1# - 3 #, #5, #6, #8, #11, *etc.*). For seven test cases (*e.g.* see test cases #6, #8, #10, #25, *etc.*), the PUH algorithm finds better solutions when $RI = 0.5$ in comparison to the condition when $RI = 0.0$ in terms of the *LL* values. Although it performs worse in terms of the *LL* values found for two test cases (see test cases #13 and #42), it is reasonable as the primary goal of the algorithm is to minimise the number of workstations. In accordance with these results, there is no doubt that the proposed algorithm finds quite promising results for the newly proposed line balancing system.

7. Conclusions and Future Work

Although there is an increasing interest in parallelisation of production lines and an increasing trend in understanding about how beneficial they are, U-shaped lines have never been parallelised in the literature so far. Neither parallel U-shaped assembly lines nor their benefits have been brought to the attention of the academia in any research. As a pioneering research in this area, this paper proposes a new line system configuration which allows utilisation of multi-line stations between the adjacent outer and inner lines while allowing the utilisation of crossover workstations between the front and back of the inner line. The problem is described clearly and its characteristics are defined explicitly. A heuristic algorithm is proposed as a possible solution approach for the suggested line system. For this aim, modifications of two well-known line balancing heuristics are combined with the understanding of favouring the most suitable tasks based on the assignment position of the lines. Briefly, tasks having more positional weights and more number of successors are given priority while assigning tasks from the front of the precedence relationships graph however tasks having less positional weights and less number of successors are prioritised when assigning tasks from the back of the precedence relationships graph.

To provide a better understanding on the advantage of the proposed parallel U-shaped assembly line system, two different examples are provided. The examples clearly illustrate the steps of the proposed algorithm and the advantages of balancing lines together. A comprehensive set of test problems available in the literature for single U-shaped lines are combined in pairs and used as test data for the proposed system and solution method (each test case is composed of a combination of two test problems). Results obtained from the proposed algorithm indicate that locating two U-shaped lines in parallel to each other helps minimise total number of required workstations. Thus, the benefit of the parallel U-shaped assembly line system over single U-shaped lines is explored and reported in this study. As the proposed algorithm finds optimal solutions for the majority of the test cases, the observed results also indicate that the proposed algorithm is competitive.

From an industrial point of view, the system proposed in this research has promising practical application opportunities in almost any sector. Production line managers who wish to establish a new

line system or to improve their existing system can make use of the parallel U-shaped assembly line system proposed in this study. In this way, companies not only save total number of operators (or workstations) but also take advantage of other benefits explained clearly in this research.

As the first research on parallel U-shaped assembly lines, this study can yield several research projects in this domain and be extended in several ways. First, an optimal solution algorithm can be developed after characterising the optimal solution properties. Second, the line system proposed in this study can be extended to the one with multiple product types. Third, resource dependent processing times can be considered for tasks performed on the lines. Finally, case studies can be conducted to show the applicability of the proposed line system.

References

[1] Miltenburg J. Balancing U-lines in a multiple U-line facility. European Journal of Operational Research. 1998;109(1):1-23.
[2] Jayaswal S, Agarwal P. Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach. Journal of Manufacturing Systems. 2014.
[3] Kim YK, Song WS, Kim JH. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers & Operations Research. 2009;36(3):853-65.
[4] Kucukkoc I, Karaoglan AD, Yaman R. Using response surface design to determine the optimal parameters of genetic algorithm and a case study. International Journal of Production Research. 2013;51(17):5039-54.
[5] Miltenburg GJ, Wijngaard J. The U-Line Line Balancing Problem. Management Science. 1994;40(10):1378-88.
[6] Urban TL. Note. Optimal Balancing of U-Shaped Assembly Lines. Management Science. 1998;44(5):738-41.
[7] Scholl A, Klein R. ULINO: Optimally balancing U-shaped JIT assembly lines. International Journal of Production Research. 1999;37(4):721-36.
[8] Erel E, Sabuncuoglu I, Aksu BA. Balancing of U-type assembly systems using simulated annealing. International Journal of Production Research. 2001;39(13):3003-15.
[9] Gökçen H, Ağpak K, Gencer C, Kizilkaya E. A shortest route formulation of simple U-type assembly line balancing problem. Applied Mathematical Modelling. 2005;29(4):373-80.
[10] Hwang RK, Katayama H, Gen M. U-shaped assembly line balancing problem with genetic algorithm. International Journal of Production Research. 2008;46(16):4637-49.
[11] Sabuncuoglu I, Erel E, Alp A. Ant colony optimization for the single model U-type assembly line balancing problem. International Journal of Production Economics. 2009;120(2):287-300.
[12] Chiang W-C, Urban TL. The stochastic U-line balancing problem: A heuristic procedure. European Journal of Operational Research. 2006;175(3):1767-81.
[13] Urban TL, Chiang W-C. An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. European Journal of Operational Research. 2006;168(3):771-82.
[14] Celik E, Kara Y, Atasagun Y. A new approach for rebalancing of U-lines with stochastic task times using ant colony optimisation algorithm. International Journal of Production Research. 2014:1-14.
[15] Toksarı MD, İşleyen SK, Güner E, Baykoç ÖF. Simple and U-type assembly line balancing problems with a learning effect. Applied Mathematical Modelling. 2008;32(12):2954-61.
[16] Kara Y, Özgüven C, Yalçın N, Atasagun Y. Balancing straight and U-shaped assembly lines with resource dependent task times. International Journal of Production Research. 2011;49(21):6387-405.

[17] Gökçen H, Ağpak K. A goal programming approach to simple U-line balancing problem. European Journal of Operational Research. 2006;171(2):577-85.
[18] Kim YK, Kim JY, Kim Y. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. European Journal of Operational Research. 2006;168(3):838-52.
[19] Kara Y, Ozcan U, Peker A. Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives. Applied Mathematics and Computation. 2007;184(2):566-88.
[20] Chiang W-C, Kouvelis P, Urban TL. Line balancing in a just-in-time production environment: balancing multiple U-lines. IIE Transactions. 2007;39(4):347-59.
[21] Kara Y, Paksoy T, Chang C-T. Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. European Journal of Operational Research. 2009;195(2):335-47.
[22] Yegul MF, Agpak K, Yavuz M. A New Algorithm for U-Shaped Two-Sided Assembly Line Balancing. Transactions of the Canadian Society for Mechanical Engineering. 2010;34(2):225-41.
[23] Özcan U, Kellegöz T, Toklu B. A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. International Journal of Production Research. 2011;49(6):1605-26.
[24] Hamzadayi A, Yildiz G. A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints. Computers & Industrial Engineering. 2012;62(1):206-15.
[25] Rabbani M, Moghaddam M, Manavizadeh N. Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout. International Journal of Advanced Manufacturing Technology. 2012;59(9-12):1191-210.
[26] Hamzadayi A, Yildiz G. A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. Computers & Industrial Engineering. 2013;66(4):1070-84.
[27] Manavizadeh N, Hosseini N-s, Rabbani M, Jolai F. 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. 2013;64(2):669-85.
[28] Battaïa O, Dolgui A. A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics. 2013;142(2):259-77.
[29] Dong J, Zhang L, Xiao T, Mao H. Balancing and sequencing of stochastic mixed-model assembly U-lines to minimise the expectation of work overload time. International Journal of Production Research. 2014:1-20.
[30] Bloomberg. A view from Toyota Caetano Portugal S.A. airport buses assembly line in Portugal plant, http://www.gettyimages.co.uk/detail/news-photo/contrac-cobus-3000-airport-buses-stand-on-the-production-news-photo/458163961. 2015.
[31] Gama. A view from Mercedes-Benz bus assembly plant in Istanbul, http://www.gama.com.tr/doc/image/projects/85/big/PfRkehaO.jpg. 2015.
[32] King-Long. A view from King Long bus assembly plant in China, http://f.etwservice.com/newsupfile/20101261358589O1.jpg. 2010.
[33] Yutong. A view from Yutong Group bus assembly line in China plant, http://www.yutong.com/english/images/news/press/2014/02/21/1249B1F58CA6D33B14C1A2DB12C9D089.jpg. 2013.
[34] Gökçen H, Ağpak K, Benzer R. Balancing of parallel assembly lines. International Journal of Production Economics. 2006;103(2):600-9.
[35] Ozcan U, Gokcen H, Toklu B. Balancing parallel two-sided assembly lines. International Journal of Production Research. 2010;48(16):4767-84.
[36] Kucukkoc I, Zhang DZ. Balancing Parallel Two-Sided Assembly Lines via a Genetic Algorithm Based Approach. Proceedings of 43rd International Conference on Computers and Industrial Engineering (CIE43). The University of Hong Kong, Hong Kong2013. p. 1-16.
[37] Kucukkoc I, Zhang DZ. A Mathematical Model and Genetic Algorithm based Approach for Parallel Two-Sided Assembly Line Balancing Problem. Production Planning & Control. 2015;doi: 10.1080/09537287.2014.994685.
[38] Kucukkoc I, Zhang DZ. Type-E Parallel Two-Sided Assembly Line Balancing Problem: Mathematical Model and Ant Colony Optimisation based Approach with Optimised Parameters. Computers & Industrial Engineering. 2015;doi: 10.1016/j.cie.2014.12.037.
[39] Kucukkoc I, Zhang DZ, Keedwell EC. Balancing Parallel Two-Sided Assembly Lines with Ant Colony Optimisation Algorithm. Proceedings of the 2nd Symposium on Nature-Inspired Computing

and Applications (NICA) at Artificial Intelligence and the Simulation of Behaviour (AISB) 2013 Convention. University of Exeter, Exeter2013. p. 21-8.
[40] Özcan U, Çerçioğlu H, Gökçen H, Toklu B. Balancing and sequencing of parallel mixed-model assembly lines. International Journal of Production Research. 2010;48(17):5089-113.
[41] Kucukkoc I, Zhang DZ. 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. 2014.
[42] Kucukkoc I, Zhang DZ. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Research. 2014;52(12):3665-87.
[43] Kucukkoc I, Zhang DZ. An Agent Based Ant Colony Optimisation Approach for Mixed-Model Parallel Two-Sided Assembly Line Balancing Problem. In: Grubbström RW, Hinterhuber HH, (Eds.) Pre-Prints of the Eighteenth International Working Seminar on Production Economics. Innsbruck, Austria: Congress Innsbruck; 2014. p. 313-28.
[44] Zhang DZ, Kucukkoc I. Balancing mixed-model parallel two-sided assembly lines. Proceedings of 2013 IEEE International Conference on Industrial Engineering and Systems Management (IESM). Rabat, Morocco: IEEE; 2013. p. 1-11.
[45] Monden Y. Toyota Production System. Norcross, GA: Industrial Engineering and Management Press; 1983.
[46] Helgeson WB, Birnie DP. Assembly line balancing using the ranked positional weight technique. Journal of Industrial Engineering. 1961;12(6):394–8.
[47] Tonge FM. Summary of a Heuristic Line Balancing Procedure. Management Science. 1960;7(1):21-42.
[48] Otto A, Otto C. How to design effective priority rules: Example of simple assembly line balancing. Computers & Industrial Engineering. 2014;69:43-52.
[49] Homepage for Assembly Line Optimization Research. ALB-Research Group.
[50] Balakrishnan J, Cheng C-H, Ho K-C, Yang KK. The application of single-pass heuristics for U-lines. Journal of Manufacturing Systems. 2009;28(1):28-40.
[51] Ağpak K, Çetinyokuş T, Gökçen H. Crossover station based simple U-type assembly line balancing. Proceedings of 5th International Symposium on Intelligent Manufacturing Systems2006. p. 118-31.
[52] Baykasoglu A. Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. Journal of Intelligent Manufacturing. 2006;17(2):217-32.

Highlights

- An innovative parallel assembly line configuration is proposed for U-shaped lines.
- Parallel U-shaped assembly line balancing problem is introduced for the first time.
- Advantages of the proposed system are presented using two numerical examples.
- Modifications of two well-known heuristics are combined as a solution method.
- Computational results demonstrate the validity of the proposed system and method.

Graphical Abstract