SIMULTANEOUS PERSPECTIVE-BASED ASSEMBLY LINE BALANCING PROBLEM

Horng-Jinh Chang[a], Chun-Hsiung Lan[b], Tung-Meng Chang[c,*]
[a] Graduate Institute of Management Sciences, Tamkang University, Tamsui, Taipei, Taiwan, R.O.C.
[b] Graduate Institute of Management Sciences, Nanhau University, Dalin, Chiayi, Taiwan R.O.C.
[c] Department of Industrial Management, Tungnan University, Taipei, Taiwan, R.O.C.
[c] Graduate Institute of Management Sciences, Tamkang University, Tamsui, Taipei, Taiwan, R.O.C.
*Correspondence: Tung-Meng Chang e-mail: tmchang@mail.tnit.edu.tw ; tmchangtn@yahoo.com.tw
Mailing address: 5F., No.79, Lane 208, Sec.3, Chengde Rd., Taipei, 10366, Taiwan, R.O.C.

Received June 2006, Accepted October 2007
No. 06-CSME-28, E.I.C. Accession 2947

ABSTRACT

Whenever a simultaneous production is allowed for the entire tasks in an assembly line, the assembly line balancing (ALB) problem becomes complicated because the simultaneous assignment is performed. In this paper, a mathematical model applying the Lingo 9.0 syntax for an advanced issue called single-model assembly line balancing problem with simultaneous production (SALBPS) is developed, and a coding technique- Three-Position Code (TPC) as well as a computerized coding program are presented to make this issue solvable. In addition, the simulated analyses of the model for various cycle times are also conducted to reveal the behavior changes of the optimal solution. The traditional method always underestimates the production rate for the SALBPS problem, but our study can more accurately estimate the system production rate for such a kind of problems. This study functions as a valuable tool because of its repeated characteristic. The proposed mathematical model and its relevant computerized coding program can make other cases easily solved by changing their input data only. Moreover, our study can help line designers quickly design or redesign the assembly line to satisfy the fluctuant environments.

Keywords: assembly line balancing; simultaneous production; SALBPS; TPC

PROBLÈME D'ÉQUILIBRAGE DE CHAÎNE D'UNE PERSPECTIVE SIMULTANÉE

RÉSUMÉ

Chaque fois qu'une production simultanée est permise pour des tâches entières d'une chaîne de montage, il en résulte un problème d'équilibrage des chaînes qui se complique en raison de la tâche simultanée effectuée. Dans le présent article, un modèle mathématique appliquant la syntaxe Lingo 9.0 à un problème avancé appelé « problème d'équilibrage des chaînes d'un modèle unique » est élaboré et un code à trois positions de technique de codage de même qu'un programme de codage informatisé sont présentés afin de pouvoir résoudre cette question. Par ailleurs, des analyses simultanées du modèle pour différents temps de cycle sont également réalisées pour démontrer les changements de comportement de la solution optimale. La méthode traditionnelle sous-estime toujours le taux de production du problème d'équilibrage, mais notre étude peut estimer de façon plus précise le taux de production du système pour ce type de problèmes. Cette étude est un outil inestimable en raison de sa caractéristique répétée. Le modèle mathématique proposé et son programme de codage informatisé pertinent permettent de résoudre facilement d'autres cas en changeant uniquement leurs données d'entrée. De plus, notre étude peut aider les concepteurs de chaînes de montage à concevoir ou reconcevoir rapidement leur chaîne pour satisfaire aux environnements fluctuants.

Mots clés : équilibrage des chaînes, production simultanée, problème d'équilibrage des chaînes, code à trois positions

NOTATIONS

1. i, $i=1,2,\dots,N$, means the i-th task of precedence diagram in an assembly line, where N is the total number of tasks on the precedence diagram.
2. K is the number of assigned workstations.
3. $\overline{K}$ means the maximum value of the number of operation units for each part; it also represents the maximum number of assignable workstations.
4. n_p is the total number of parts (paths) on the precedence diagram.
5. IP is the set of all arcs (i,j) on the precedence diagram.
6. $I_i=\{m_1^i,m_2^i,\dots,m_r^i\}$, where m_r^i means one Three-Position Code for the i-th task.
7. CT is the cycle time of the assembly line.
8. A_{pk} is the set of tasks belonged to the p-th path assigned to the k-th workstation, where $p=1,2,\dots,n_p$, $k=1,2,\dots,K$.
9. $t_{m_r^i}$ is the processing time of task i with Three-Position Code m_r^i.
10. IR is the total idle rate of the production system.
11. IT means the sum of idle time of all workstations.
12. $$x_{m_r^i k}=\begin{cases}1 & \text{if task } i \text{ with Three-Position Code } m_r^i \text{ is assigned to workstation } k;\\ 0 & \text{otherwise.}\end{cases}$$

1. INTRODUCTION

Assembly line balancing (ALB) is probably one of the oldest problems in industry, and an important subject of many production systems. ALB is to assign tasks to workstations as evenly as possible so that the workstation times will be as equal as possible [1]. When there is only one model of a product that is being assembled on the line, such a problem is called the single-model assembly line balancing problem (SALBP). During the design of the line, all tasks to be done, the times required performing each task, and the constraints indicating the order in which may be done (task precedence diagram) are analyzed. Based on this analysis the SALBP can be formulated in two forms, (1) the single-model assembly line balancing problem type 1 (SALBP-1) is to minimize the number of workstations for a given cycle time; (2) the single-model assembly line balancing problem type 2 (SALBP-2) is to minimize the cycle time for a given number of workstations.

Based on literatures, the philosophy of concurrent engineering applying to the ALB problems with considering the simultaneous production is seldom mentioned. Whenever simultaneous production is allowed for the entire tasks in an assembly line, the ALB problem become hard solving because the simultaneous assignment is performed. In this paper, a mathematical model for solving the single-model assembly line balancing problems with simultaneous production (SALBPS) is developed. In order to solve such a complicated problem, a coding technique, Three-Position Code (TPC), is proposed to re-code the tasks of assembly line, and a computerized coding program written in C++ to automatically generate those TPCs is also provided.

In the past several decades, many researchers have studied the single-model ALB problem. Klein and Scholl [2] presented a new branch and bound approach, called SALOME-2, to solve SALBP-2. Sarin et al. [3] developed a heuristic enumeration method for the single-model, stochastic assembly line balancing problem. Ugurdag et al. [4] formulated an integer-programming model for SALBP-2, and used a two-stage heuristic procedure to solve this problem. Sabuncuoglu et al. [5] proposed an efficient genetic algorithm to solve the deterministic and single-model ALB problem. Nicosia et al. [6] considered a dynamic programming algorithm and some fathoming rules to balance assembly lines with different workstations. Fleszar and Hindi [7] presented a new heuristic algorithm and new reduction techniques for the SALBP-1. Agpak and Gokcen et al. [8] proposed a 0-1 integer-programming model to establish balance of the assembly line with minimum number of workstations and resources. For single-model assembly line balancing problems, a large number of exact and heuristic procedures are available (see Erel and Sarin [9]; Rekiek et al. [10]; Scholl and Becker [11]. Despite some progress in exact and heuristic methods to solve ALB problems, software implementing those methods is still the most commonly used tool in industry.

According to the above-mentioned researches, the assignment way of tasks into workstations for SALBP is in a sequential assignment way, and none of them apply the philosophy of concurrent engineering in considering the simultaneous production. The concurrent engineering (CE) is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support [12]. Lettic et al. [13] applied concurrent engineering to develop a paper-based workbook style methodology, Herder and Weijnen [14] used it to explore the external factors of chemical design, Duffy and Salvendy [15] provided a connection between CE and virtual reality for human resource planning, and Finea et al. [16] also presented a goal-programming modeling approach to address three-dimensional concurrent engineering (3D-CE) problems involving product, process and supply chain design. To the best of our knowledge, there is no one applying CE to the ALB problems.

Line designers are responsible for economical production by determining the manufacturing processes and the machine tools to be used [17]. So, it is important for designers to balance workstations of an assembly/ production line in order to obtain an economical production. Stevenson [18] stated that the ALB efficiency increases if the total idle rate decreases. In addition, while the minimum total idle time is the main goal for the ALB problem, the spirit of Just-in-Time (JIT) philosophy is mainly pursued. Fullerton et al. [19] stated that the most consistent benefit from Just-in-Time philosophy adoption found in the empirical studies is to reduce inventory levels and waiting (idle) time.

In summary, this work integrates SALBP with considering the simultaneous production to form a new issue called single-model assembly line balancing problem with simultaneous production, abbreviated as SALBPS. And, a coding technique (TPC) and its computerized coding program are provided to facilitate this problem solvable.

The paper is organized as follows. Section 2 presents a brief description of the problem and provides a coding technique to deal with the simultaneous production in Lingo software. The assumptions and the model formulation are described and developed in Section 3. An illustrative example explaining how to solve this issue and the simulated analyses for various cycle times are conducted in Section 4. Finally, some conclusions are given.

2. PROBLEM DESCRIPTION AND CODING SYSTEM

2.1 Problem Description

The objective of this work is to minimize the total idle rate (or maximize the line balancing efficiency) for an ALB problem with considering simultaneous production under a given cycle time. This paper considers the situation of several different parts to be produced simultaneously (a product consists of these parts) in an assembly line, and each part has its routine tasks to be performed. Different parts may require the same task to be made during their manufacturing, and thus the manufacturing flow of these parts can be linked as a graph-type production line. The precedence relations of these tasks can be drawn as an acyclic precedence diagram, where the nodes (tasks) on the diagram are numbered according to a topological ordering and the processing time of each task is constant. That is $i < j$ for all arcs (i, j) which means that task i has to be completed before task j, and the processing time of each task is known.

Besides, an assembly line consists of a sequence of workstations. The amount of time available at each workstation is called cycle time and it is predetermined by the desired production rate. The difference between the cycle time and the sum of the task times assigned at any workstation is called workstation's idle time. Hence, the total idle time of an assembly line is the sum of workstation's idle time over all workstations, and the total idle rate is a ratio of the total idle time to the time available in the line. Because the tasks belonging to a part (a path) assigned into a workstation are performed by one operator, the tasks belonging to different parts can be assigned simultaneously into a workstation when the task assignments are performed. Under considering simultaneous production, the objective is to assign tasks in such a manner so as to minimize the number of workstations on the assembly line, without having the task time assigned at any workstation exceed the cycle time and without violating the precedence constraints. Minimizing the number of workstations is equivalent to minimizing the total idle rate of the assembly line.

2.2 Coding System

In order to solve the SALBPS problem in Lingo syntax, a coding technique, Three-Position Code (TPC), is proposed to re-code the tasks on the precedence diagram. A TPC uses three positions (G, P, O) to represent each task on the precedence diagram. For a task code (G, P, O), G shows the group number where such a given task should be assigned; P indicates the path number where a given task should be belonged; O stands for the operation order number of a given task in its specific group and path, where the groups count from the number of source nodes on the precedence diagram, and the paths are determined by the number of parts on assembly line. A path may consist of tasks in sequence in one workstation, and it indicates the partial manufacturing process of one part. In addition, a "Codes Generator" developed in C++ language is provided to automatically generate TPCs for each task in this work. Because there may have some same tasks among different parts, the proposed coding technique (TPC) probably leads to a task to own one more codes, and these codes have to be assigned into the same workstation during assignment.

Moreover, a flow chart (shown in figure 1) to explain the solving steps of single-model assembly line balancing problem with simultaneous production (SABLPS) is provided. The main steps of this flow are as follows.

Step 1. Generate the precedence matrix from the precedence diagram. The precedence matrix is an upper-triangular matrix with an *ab*-th entry of 1 if task *a* precedes immediately task *b*; otherwise is zero [20]. The precedence matrix can be generated from the precedence diagram.

Step 2. Input the generated matrix into the "Codes Generator". Put this generated matrix into the "Codes Generator" to generate the Three-Position Code (TPC) for each task of all parts, where TPC is provided for each task on the precedence diagram to make the SALBPS problem easily solved in Lingo syntax.

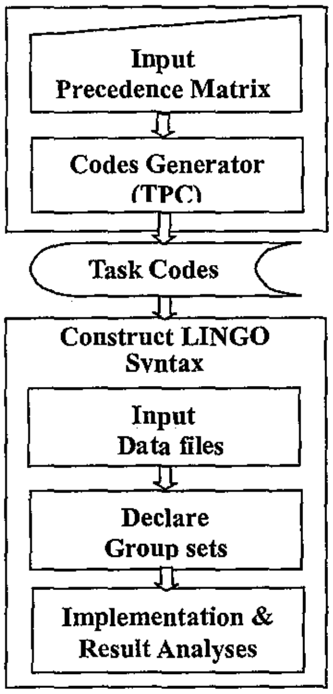

Figure 1. Flow chart for implementing LINGO modeling

Step 3. Construct the model in Lingo syntax. After TPCs of tasks are generated, the steps of constructing Lingo program for the proposed model can be conducted. They are described as follows.

3.1 Input the Data files. The processing time of each task, the cycle time of the assembly line, and other related data are functioning as data files for Lingo program.

3.2 Declare the Group sets. The group categories, TPCs of tasks, precedence relations of tasks, and different TPCs but representing the same tasks from different paths (parts) are declared as the Group sets.

3.3 Construct the model. Based on the mathematical model, construct the model by applying the syntax of Lingo 9.0 extended version.

Step 4. Implement the Lingo and analyze the results. When all related data are inputted, run the Lingo software to solve the problem and analyze the results.

3. ASSUMPTIONS AND MODEL FOMULATION

3.1 Assumptions

The assumptions of the model are listed below:

1. The tasks of all parts of a product can be expressed as a precedence diagram, and the parts can be produced simultaneously in an assembly line.

2. The processing time of each task is known constant.
3. The same tasks belonged to different parts must be assigned to the same workstations and have the same processing times.
4. WIP (work in process) inventory buffer is not allowed between workstations.

3.2 Model Formulation

The mathematical model presenting for solving the single-model assembly line balancing problem with simultaneous production (SALBPS) is constructed as follows.

$$\min_{K, x_{m_r^i k}} \quad IR = \frac{IT}{n_p \times CT \times K} \tag{1}$$

$s.t.$

$$\sum_{k=1}^{K} x_{m_r^i k} = 1 \qquad \forall m_r^i,\ m_r^i \in I_i, i = 1,2,\ldots,\ \mathrm{N} \tag{2}$$

$$\sum_{k=1}^{K} (kx_{m_r^i k} - kx_{m_s^i k}) = 0 \qquad \forall m_r^i, m_s^i \in I_i, i = 1,2,\ldots,\mathrm{N} \tag{3}$$

$$0 \le \sum (kx_{m_r^j k} - kx_{m_r^i k}) \le 1 \qquad \forall (i, j) \in IP \tag{4}$$

$$\sum_{\substack{m_r^i \in I_i \\ i \in A_{pk}}} t_{m_r^i} x_{m_r^i k} \le CT \qquad \forall i, p, k \tag{5}$$

$$K \le \overline{K} \tag{6}$$

$$x_{m_r^i k} \in \{0, 1\} \qquad \forall i, k \tag{7}$$

The objective function of the SALBPS model is to minimize the total idle rate of the entire assembly line as given in Eq. (1), where $IT = \sum_{k=1}^{K} \sum_{p=1}^{n_p} (CT - \sum_{\substack{m_r^i \in I_i \\ i \in A_{pk}}} t_{m_r^i} x_{m_r^i k})$. The calculation for time available in the assembly line is different from that in the traditional assembly line. It needs to consider the number of parts simultaneously produced on the assembly line. Eq. (2) guarantees that every task (task code) is assigned into at most one workstation. It is a necessary constraint while modeling the ALB problems. Eq. (3) enforces that different TPCs from the same task belonging to different paths (parts) have to be assigned to the same workstation. Eq. (4) represents task *j* has to assign to the same workstation of task *i* or to assign to the next workstation of task *i*, where *i* is the precedence task of *j*. This technological constraint indicates that the precedence relationships of all tasks have to be observed. Eq. (5) means that the sum of processing time of tasks for each path (part) assigned into any workstation cannot be greater than the cycle time. It is different from the traditional ALB model requiring the sum of time of all tasks assigned into any workstation cannot be greater than the cycle time. Eq. (6) limits that the assigned number of workstations cannot be greater than the maximum number of assignable workstations. The last equation, Eq. (7), shows that $x_{m_r^i k}$ are binary variables. In addition, $x_{m_r^i k}$ and K are decision variables of the proposed model, and $t_{m_r^i}$, CT, $\overline{K}$, and n_p are given parameters. This model is categorized as an Integer Nonlinear Programming (INLP) problem in Lingo 9.0 extended version.

4. ILLUSTRATIVE EXAMPLE

This section provides an illustrative example showing how to use coding technique (TPC) to solve the SALBPS problem. The product in this illustrative example is made up of ten different parts (paths) to be produced simultaneously in an assembly line; each part has seven different tasks to be performed. The precedence diagram can be expressed as figure 2; the numbers above the nodes represent task processing times. It can be divided into three groups (source nodes), the first group includes six different paths (parts), the second group has three different paths, and the last group owns only one path. Totally, there are ten paths (parts) and seven total operation orders for each part in this case (listed in table 1). In table 1, the paths consisting of tasks 1, 4, 7, 11, 15, 17, 19, and tasks 1, 4, 8, 12, 15, 17, 19 mean the manufacturing route of parts 1 and 2, respectively. There have some same tasks 1, 4, 15, 17, and 19 between the two different parts.

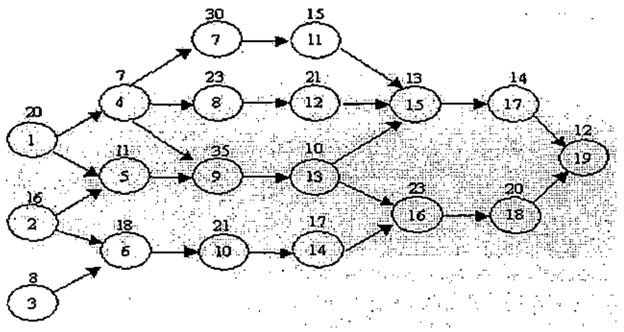
Figure 2. Illustrative example for a precedence diagram

Table 1. The total paths on the precedence diagram shown in figure 2

Group	Path	Task Unit (in order)						
1	1	1	4	7	11	15	17	19
	2	1	4	8	12	15	17	19
	3	1	4	9	13	15	17	19
	4	1	4	9	13	16	18	19
	5	1	5	9	13	15	17	19
	6	1	5	9	13	16	18	19
2	7	2	5	9	13	15	17	19
	8	2	5	9	13	16	18	19
	9	2	6	10	14	16	18	19
3	10	3	6	10	14	16	18	19

The processing procedures for this example are described as follows. First, generate a precedence matrix from the precedence diagram. The precedence matrix generated from the precedence diagram is shown in figure 3. Second, input the precedence matrix generated from the precedence diagram into the "Codes Generator" to automatically create the TPCs of each task. The code of every task generated from "Codes Generator" is summarized in table 2. The TPC of task 8 shows (1, 2, 3), where 1 means the first group category, 2 means the second path (part), and 3 means the third operation order in the second part (shown in table 2). In addition, the TPCs (2, 3, 2) and (3, 1, 2) both represent task 6 and they belong to part 9 and part 10, respectively (shown in table 2). Third, construct the proposed model in Lingo syntax. While the entire TPCs of every task are generated, the proposed mathematical model can be expressed and solved by the syntax of Lingo 9.0 extended version. The steps of constructing

$$PM = \begin{bmatrix}
- & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & - & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & - & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & - & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & - & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & - & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & & & - & 0 & 1 & 1 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & - & 0 & 1 & 0 & 0 & 0 \\
 & & & & & & & & & & & & & & - & 0 & 1 & 0 & 0 \\
 & & & & & & & & & & & & & & & - & 0 & 1 & 0 \\
 & & & & & & & & & & & & & & & & - & 0 & 1 \\
 & & & & & & & & & & & & & & & & & - & 1 \\
 & & & & & & & & & & & & & & & & & & -
\end{bmatrix}$$

Figure 3. The precedence matrix of the precedence diagram

Table 2. The codebook of each task for the proposed illustration

Group	Task	Code	Group	Task	Code	Group	Task	Code
1	1	(1,1,1)	1	15	(1,1,5)	2	6	(2,3,2)
		(1,2,1)			(1,2,5)		9	(2,1,3)
		(1,3,1)			(1,3,5)			(2,2,3)
		(1,4,1)			(1,5,5)		10	(2,3,3)
		(1,5,1)		16	(1,4,5)		13	(2,1,4)
		(1,6,1)			(1,6,5)			(2,2,4)
	4	(1,1,2)		17	(1,1,6)		14	(2,3,4)
		(1,2,2)			(1,2,6)		15	(2,1,5)
		(1,3,2)			(1,3,6)		16	(2,2,5)
		(1,4,2)			(1,5,6)			(2,3,5)
	5	(1,5,2)		18	(1,4,6)		17	(2,1,6)
		(1,6,2)			(1,6,6)		18	(2,2,6)
	7	(1,1,3)		19	(1,1,7)			(2,3,6)
	8	(1,2,3)			(1,2,7)		19	(2,1,7)
	9	(1,3,3)			(1,3,7)			(2,2,7)
		(1,4,3)			(1,4,7)			(2,3,7)
		(1,5,3)			(1,5,7)	3	3	(3,1,1)
		(1,6,3)			(1,6,7)		6	(3,1,2)
	11	(1,1,4)	2	2	(2,1,1)		10	(3,1,3)
	12	(1,2,4)			(2,2,1)		14	(3,1,4)
	13	(1,3,4)			(2,3,1)		16	(3,1,5)
		(1,4,4)		5	(2,1,2)		18	(3,1,6)
		(1,5,4)			(2,2,2)		19	(3,1,7)
		(1,6,4)						

the proposed mathematical model by the Lingo syntax are conducted as follows. (1) Input all related data, such as the number of groups, paths, and total operation orders and the upper bound of the number of workstations available. The processing time of each task and the desired cycle time for an assembly line are also given. (2) Declare all required information. The code of each task and the precedence relations between task units are defined and declared by applying the Three-Position Code generated from the "Codes Generator". Then, the code set of each task is declared. Each code set consists of different TPCs but representing the same task. For example, TPCs in the I_2 set, where $I_2=\{(2,1,1), (2,2,1), (2,3,1)\}$ (shown in table 2), all represent the same task (task 2). The proposed model formulation expressed by the syntax of Lingo 9.0 extended version through the TPCs is shown in Appendix A, and the built-in Global Solver is selected as solving method because the model is an integer nonlinear programming problem. According to the statistics function of Lingo 9.0, the above model formulation has totally 460 variables and 517 constraints, including 350 integer variables, 102 nonlinear variables, and 110 linear inequalities, 346 linear equalities, 61 nonlinear constraints, respectively.

The problem is solved using Lingo 9.0 software on a 1.0 GHz personal computer. The optimal assignments are shown in table 3. Five workstations are utilized with a cycle time 35 minutes in the optimal solution; the total idle rate associated with this example is 0.321. In the figure 4, it shows that the tasks belonging to different parts to be produced simultaneously in an assembly line are assigned into each workstation. Tasks 1, 2, 3, 4, 5, and 6 are assigned into workstation 1(shown in table 3); they can be separately and simultaneously produced by different operators in workstation 1. The minimal number of workstations means that the total idle rate resulting from such a solution is minimum and the number of assigned workstations is also kept as minimum as possible.

Table 3. Optimal assignments of tasks to workstation

Workstation	Assignment of tasks
1	1 2 3 4 5 6
2	7 8 9 10
3	11 12 13 14 15
4	16 17
5	18 19

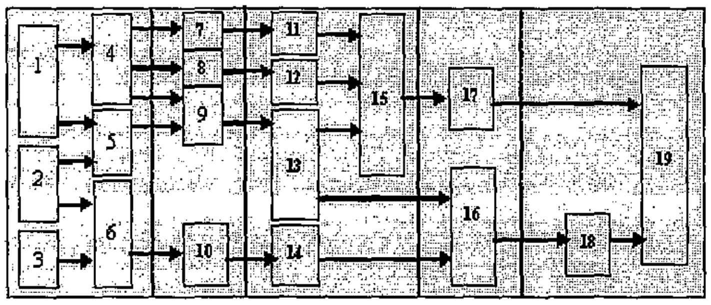

Figure 4. The diagram of the optimal workstation assignments for all tasks

In the ALB literatures, two conditions are generally accepted. First, the desired range of the number of workstations should be $1 \le K \le \overline{K}$, and it usually can be set by management with the consideration of floor space, budget, and operational limitations. Second, the desired range of a cycle time should be $t_{\max} \le CT \le \overline{CT}$, where $t_{\max}$ and $\overline{CT}$ mean the longest task processing time among all tasks and a given upper bound on cycle time, respectively. Generally, the processing time of tasks can be determined by using time study techniques [21] and the cycle times are related to the product demand of market.

In this illustrative example, the $t_{\max}$ is 35 minutes, and the allowed largest cycle time is 133 minutes when all tasks are assigned into one workstation. $\overline{K}$ is 7 when one task for each part is assigned into one workstation. So, different cycle times (from 35 minutes to 135 minutes) are conducted to simulate the changes of the optimal solution. The detailed results are shown in table 4 and summarized as follows:

1. The maximum assigned number of workstations is constrained by the maximum value of the number of operation units for each part. Besides, each given cycle time would obtain an optimal number of assigned workstations and this optimal number of assigned

workstations should be less than the $\overline{K} = 7$.

2. In general, the optimal assigned number of workstations and the total idle rate are decreasing if the cycle time is increasing.
3. For the same assigned optimal number of workstations during the simulation, the total idle rate increases whenever the cycle time increases.
4. The total idle rate, 0.207, is minimization while the optimal assigned number of workstations (K) equals to 2 and the cycle time (CT) is given as 75 minutes. More specifically, the production rate of this line design is maximal.

Table 4. The optimal solution for various cycle times and workstations

CT \ K	1	2	3	4	5	6	7
35	-	-	-	-	.321	-	-
40	-	-	-	.257	-	-	-
45	-	-	-	.339	-	-	-
50	-	-	-	.406	-	-	-
55	-	-	.279	-	-	-	-
60	-	-	.339	-	-	-	-
65	-	-	.390	-	-	-	-
70	-	-	.434	-	-	-	
75	-	.207	-	-	-	-	-
80	-	.257	-	-	-	-	-
85	-	.301	-	-	-	-	-
90	-	.339	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
135	.119	-	-	-	-	-	-

5. CONCLUSIONS

In recent researches, most researchers explored non-synchronous production models because considering simultaneous production perspective always makes the problem complicated. The simultaneous assignment way can be handled when different parts of a product can be produced simultaneously in an assembly line. In this paper, we proposed a way to make the above-mentioned problem solvable. Followings are the major contributions of this study.

1. This study mainly focuses on solving the assembly line balancing problem with considering simultaneous production (SALBPS) by constructing a specific mathematical model and the relevant techniques for solving such a complicated issue.
2. The syntax of packaged software Lingo 9.0 is applied to describe the proposed specific

mathematical model for coping with the SALBPS problems. In addition, the proposed mathematical model owns the repeated characteristic because of its solving method, built-in "Global Solver" in Lingo 9.0, can make other cases easily solved by changing their input data only. Moreover, our study can help line designers quickly design or redesign the assembly line to satisfy the fluctuant environments as well as the simultaneous production considerations.

3. This paper presents a coding technique called TPC to successfully link our mathematical model to solve the SALBPS problems. In addition, this paper also proposes a computerized coding technique called Codes Generator in order to automatically generate the TPCs for each task.
4. The traditional method to determine the cycle time (production rate) for the SALBPS problem would be overestimation (underestimation) if the simultaneous production situation exists. Thus, our study can function as a more accurately measure to estimate the system production rate for an assembly line balancing problem.

In summary, though this paper only considers the single-model assembly line balancing problem with considering simultaneous production, a future focal research is strongly recommended to explore more realistic industry need, such as the mixed–model or multi–model assembly line balancing problem with the simultaneous production perspective.

ACKNOWLEDGEMENTS

We would like to thank the anonymous referees for their constructive comments. The comments have greatly improved the quality of the paper.

REFERENCES

1. Suer, G.A., Designing parallel assembly lines, Computers ind. Engng, Vol. 35, 1998, p467-470.
2. Klein, R. and Scholl, A., Maximizing the production rate in sample assembly line balancing- A branch and bound procedure, European Journal of Operational Research, Vol. 91, 1996, p367-385.
3. Sarin, S.C., Erel, E. and Dar-El, E.M., A methodology for solving single-model, stochastic assembly line balancing problem, Omega, Int. J. Mgmt. Sci., Vol. 27, 1999, p525-535.
4. Ugurdag, H. F., Rachamadugu, R. and Papachristou, C.A., Designing paced assembly lines with fixed number of stations, European Journal of Operational Research, Vol. 102, 1997, p488-501.
5. Sabuncuoglu, I., Erel, E. and Tanyer, M., Assembly line balancing using genetic

algorithms, Journal of Intelligent Manufacturing, Vol. 11, 2000, p295-310.

6. Nicosia,G., Pacciarelli, D. and Pacifici, A., Optimally balancing assembly lines with different workstations, Discrete Applied Mathematics, Vol. 118, 2002, p99-113.
7. Fleszar, K. and Hindi, K.S., An enumerative heuristic and reduction methods for the assembly line balancing problem, European Journal of Operational Research, Vol. 145, 2003, p606-620.
8. Agpak, K. and Gokcen, H., Assembly line balancing: Two resource constrained cases, Int. J. Production Economics, Vol. 96, 2005, p129-140.
9. Erel, E. and Sarin, S.C., A survey of the assembly line balancing procedures, Production Planning and Control, Vol. 9, 1998, p414-434.
10. Rekiek, B., Dolgui, A., Delchambre, A. and Bratcu, A., State of art of optimization methods for assembly line design, Annual Reviews in Control, Vol. 26, 2002, p163-174.
11. Scholl, A. and Becker, C., State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, European Journal of Operational Research, Vol. 168, 2006, p666-693.
12. Prasad, B., Concurrent Engineering Fundamentals, Prentice-Hall Engewood Cliffs N. J., 1996.
13. Lettice, F., Smart, P. and Evans, S., A workbook-based methodology for implementing concurrent engineering, International Journal of Industrial Ergonomic, Vol.16, 1995, p339-351.
14. Herder, P.M. and Weijnen, M.P.C., A concurrent engineering approach to chemical process design, Int. J. Production Economics, Vol. 64, 2000, p311-318.
15. Duffy, V.G. and Salvendy, G., Concurrent engineering and virtual reality for human resource planning, Computers in Industry, Vol. 42, 2000, p109–125.
16. Finea, C.H., Golany, B. and Naseraldin, H., Modeling trade offs in three-dimensional concurrent engineering: a goal programming approach, Journal of Operations Management, Vol. 23, 2005, p389–403.
17. Cogun C., A correlation between deviations in form and size tolerances, Transactions of the Canadian Society for Mechanical Engineering, Vol. 13, 1989, p75-78.
18. Stevenson, W.L., Operations Management, 8th ed., McGraw-Hill N. Y., 2002.
19. Fullerton, R.R., McWatters, C.S. and Fawson, C., An examination of the relationships between JIT and financial performance, Journal of Operations Management, Vol. 21, 2003, p383–404.
20. Hoffmann, T.R. Assembly line balancing with a precedence matrix, Management Science, Vol. 9, 1963, p551-562.
21. Chi Leung, PH. and Sau Fun, FN., A study of the effect of time variations for assembly

line balancing in the clothing industry, International Journal of Clothing Science and Technology, Vol. 11, 1999, p181-188.

APPENDIX A

Model formulation of the example problem in Lingo syntax

```
MODEL:
DATA:
  (omitted)
ENDDATA
SETS:
  STATION/1..NU_WS/:TIDLE;
  GROUP/1..NU_GROUP/;
  PATH /1..NU_PATH/;
  UNIT/1..NU_UNIT/;
  Z(GROUP,PATH)
     /1,1 1,2 1,3 1,4 1,5 1,6 2,1 2,2 2,3 3,1/;
  TASK(GROUP,PATH,UNIT)
     /1,1,1 1,2,1 1,3,1 1,4,1 1,5,1 1,6,1 2,1,1 2,2,1 2,3,1 3,1,1
      1,1,2 1,2,2 1,3,2 1,4,2 1,5,2 1,6,2 2,1,2 2,2,2 2,3,2 3,1,2
      1,1,3 1,2,3 1,3,3 1,4,3 1,5,3 1,6,3 2,1,3 2,2,3 2,3,3 3,1,3
      1,1,4 1,2,4 1,3,4 1,4,4 1,5,4 1,6,4 2,1,4 2,2,4 2,3,4 3,1,4
      1,1,5 1,2,5 1,3,5 1,4,5 1,5,5 1,6,5 2,1,5 2,2,5 2,3,5 3,1,5
      1,1,6 1,2,6 1,3,6 1,4,6 1,5,6 1,6,6 2,1,6 2,2,6 2,3,6 3,1,6
      1,1,7 1,2,7 1,3,7 1,4,7 1,5,7 1,6,7 2,1,7 2,2,7 2,3,7 3,1,7/:TIME;
  PRED(TASK,TASK)/ omitted/;
  TXS(TASK,STATION):X;
  TYS(Z,STATION):Y,IDLE;
     .
     .
     .
ENDSETS
MIN = TOTAL/TCT;
TCT = N * CT * NU_PATH;
TOTAL = @SUM(STATION(K):TIDLE(K));
@FOR (TASK(A,B,I)| I # EQ # NU_UNIT:N = @MAX(STATION(K):(K*X(A,B,I,K))));
@FOR(TASK(A,B,I):@SUM(STATION(K):X(A,B,I,K))=1);
```

```
@FOR(CTASK(R,A,B,C,D,E,F):@SUM( STATION( K):K * X( A,B,C,K) - K * X( D,E,F,K)) = 0);
@FOR(PRED(A,B,I,A,B,J):@SUM(STATION(K):K * X(A,B,J,K) - K * X(A,B,I,K)) <= 1;
          @SUM(STATION( K): K * X(A,B,J,K) - K * X(A,B,I,K)) >= 0);
@FOR(STATION(K):@FOR( Z(A,B):@SUM(TXS(A,B,I,K): TIME( A,B,I) * X( A,B,I,K)) <= CT));
@FOR (STATION(K):@FOR (Z(A,B):
          CT - @SUM(TASK(A,B,I):TIME(A,B,I)* X(A,B,I,K ))=IDLE(A,B,K)));
@FOR (STATION(K):@SUM(Z(A,B):Y(A,B,K)) = TIDLE(K));
@FOR (TYS(A,B,K):Y(A,B,K)=@IF(IDLE(A,B,K)# EQ # CT,0,IDLE(A,B,K)));
@FOR(TXS:@BIN(X));
END
```