Journal of the Korean
OR/ MS Society
Vol. 19, No. 2, August 1994

A New Mathematical Formulation For The Classical Assembly Line Balancing Problem

Dooyoung Shin* and Daeyong Lee**

Abstract

This paper presents a new integer formulation (Type Ⅲ ALB) for a single model assembly line balancing problem. The objective of the formulation is to minimize the total idle time, which is defined as the product of the number of work stations and the cycle time minus the total work content. This formulation considers currently existing Type I (minimizing the number of work stations for a given cycle time) and type Ⅱ (minimizing the cycle time for a given number of work stations) formulations as its special cases and provides the global minimum solutions of the cycle time and the number of work stations. This information would be of great value to line designers involved in designing new assembly lines and rebalancing old lines under flexible conditions. Solution methods based on combination of Type I and Type Ⅱ approaches are also suggested and compared.

1. Introduction

The main objective of a classical single model assembly line balancing (ALB, hereafter) is to minimize the idle time by finding an optimal mix of a cycle time and the number of work stations along the line. Even though this objective has been well understood and widely accepted (see Kilbridge and Wester [11]), it has never appeared in the form of the objective function in integer formulations. Instead, the idle time was minimized, either by minimizing cycle time by fixing the number of work stations (Type Ⅱ ALB) (see Baybars [2]). Although in the ALB literature only Type I and Type Ⅱ problems have been considered, these formulations do not provide the minimum idle time over the feasible ranges of cycle time and the number of work stations.

* Department of Management, College of Business, Mankato State University, Mankato, MN 56001
** Department of Business Administration, College of Business, Chosun University

When a new assembly line is designed or when a current assembly line is rebalanced, an optimal mix of the number of work stations and the cycle time is found so as to minimize line installation costs or to maximize production rate. Especially, the adoption of Just-in-time (JIT) production system often necessitates the more frequent line-rebalancing (usually once a month) that requires manufacturing firms to quickly accomodate unpredictable changes in production rates (cycle times), number of work stations (assembly workers), line layouts, and so on. The traditional Type I and Type II ALB approaches, however, do not address these changes properly because it fails to recognize the dynamic and flexible nature of JIT. Since the main objective of the ALB problem is to minimize the idle time, it may be more desirable to consider both the cycle time and the number of work stations as decision bariables in 0-1 formulations where the objective is to minimize the idle time.

While a few authors have considered cycle time and the number of work stations as variables, the objective of minimizing the idle time directly has not been considered. Rosenblatt and Carlson [3] have proposed a profit maximization ALB model with a solution procedure which maximizes the profit but not the efficiency of an assembly line. Deckro [4] has suggested a minimization model in which a weighted objective (typically, cost function) is minimized. As is indicated by Rosenblatt and Carson, however, profit maximization or cost minimization does not necessarily minimize the idle time which is a primary measure of the efficiency of an assembly line.

In this paper, an attempt will be made to formulate ALB problems using an integer programming formulation, so that the optimal mix of the number of work stations and the cycle time can be obtained. The initial form of the formulation would be nonlinear. Later in the Appendix, however, we will discuss how we can convert the nonlinear formulation into a linear integer formulation. The nonlinear formulation considers Type I and Type II ALB problems as its special cases. The purpose of the objective function is to minimize the total idle time along the assembly line by minimizing the product of the number of work stations and the cycle time. We shall refer to this problem as Type III ALB problem. Later in the paper, solution methods for solving Type III problems which are based on currently existing Type I and II solution approaches will be suggested.

2. Preliminaries

We note that the following are assumed to be true for our ALB problems throughout the

paper. Furthermore, we also note that our discussion is confined to single-model ALB problems.

1. Demand is known with certainty.
2. The basic manufacturing method is unalterable.
3. Each task is considerd indivisible.
4. No uncertain machine breakdowns, equipment failures and worker allowances are assumed.
5. All work stations under consideration are identical.

To faciiitate the exposition, we employ the following notational conventions throughout the paper.

c : cycle time

$\bar{c}$: a known upper bound on cycle time ($\bar{c}$ is determined on the basis of a target production rate)

$\underline{c}$: a known lower bound on cycle time ($\underline{c}$ is determined on the basis of a target production rate)

r : a predetermined production rate

N : number of tasks to be assigned

t_i : processing time of i-th task

t_{max} : $\mathrm{Max}\{ t_i \}$, $i = 1, \cdots, N$

$[x]^+$: the smallest integer greater than or equal to x

$|y|$: number of elements in y

m : number of work station

$\underline{m}$: a (known) lower bound on the number of work stations required

$\underline{m} = [\sum_{i=1}^{N} t_i / c]^+$ if c is unknown.

$\bar{m}$ = a (known) upper bound on the number of work stations.

In general, m can be found by using a heuristic method or by using a trial directive.

G : $\{(a,b) |$ task b immediately follows task a$\}$

P_i : the set of tasks which must precede task i

S_i : the set of tasks which must succeed task i

F : the set of tasks with no succeeding tasks.

E_i : the earliest work station to which task i can be assigned.

$$E_i = \begin{cases} 1, \text{ if } P_i = \{ \Phi \} \\ [(t_i + \sum_{k \in P_i} t_k) / \bar{c}\ ]^+, \text{ otherwise} \end{cases}$$

L_i : the latest work station to which task i can be assigned.

$$E_i = \begin{cases} m, \text{ if } S_i = \{\Phi\} \\ m_1 + 1 - [t_i + \sum_{k \in S_i} t_k) / \bar{c}\ ]^+, \text{ otherwise} \end{cases}$$

$$X_{ij} = \begin{cases} 1, \text{ if task i is assigned to station j.} \\ 0, \text{ otherwise} \end{cases}$$

$$Y_i = \begin{cases} 1, \text{ if station j is needed} \\ 0, \text{ otherwise} \end{cases}$$

T_j : subset of all tasks that can be assigned to station j,
$j = 1,\cdots,\bar{m}$, without violating precedence relations.

W_i : subset of all work stations that can accept task i,
$i = 1,\cdots,N$. (i. e. , $Wi = \{j \mid i \in T_j\}$)

T_j and Wi can be determined on the basis of the notion of 'early' and 'late' work stations for each task (i. e. , Ei and Li) introduced by patterson and Albracht [5]. Although the computation of 'early' and 'late' stations requires a known value of c, a known upper bound $\bar{c}$ can be used to reduce the number of variables.

3. Type Ⅲ ALB Formulation

we first propose the following integer nonlinear formulation of the ALB problem as Type Ⅲ ALB problem. This formulation considers Type I and Type Ⅱ ALB problems as its special cases.

Problem(P−0)

$$\underset{(c,x,y)}{\text{Min}} \quad c\left(\sum_{j=1}^{m} y_j\right) - \sum_{i=1}^{N} t_i \tag{1}$$

s.t.
$$\sum_{j \in W_i} x_{ij} = 1, \qquad i = 1,\cdots,N \tag{2}$$

$$\sum_{i \in T_j} t_i x_{ij} \le c, \qquad j = 1,\cdots,\bar{m} \tag{3}$$

$$\sum_{j \in W_a} j x_{aj} \le \sum_{k \in W_b} k x_{bk} \qquad (a,b) \in G \tag{4}$$

$$\sum_{i\in T_j} x_{ij} \le |T_j|, \qquad j=1,\cdots,\underline{m}$$
$$\sum_{i\in T_j} x_{ij} \le |T_j|\, y_j, \qquad j=\underline{m}+1,\underline{m}+2,\cdots,\bar{m} \tag{5}$$

$$y_{j+1} \le y_j, \qquad j=\underline{m}+1,\underline{m}+2,\cdots,\underline{m}-1 \tag{6}$$

$$\underline{c} \le c \le \bar{c} \tag{7}$$

$$x_{ij} = 0 \text{ or } 1, \tag{8}$$

$$y_{ij} = 0 \text{ or } 1. \tag{9}$$

The purpose of objective function (1) is to minimize the total idle time, which is equivalent to the difference between the product of the cycle time and the number of work stations and the sum of total processing times which is a constant. Constraint (2) implies that every task should be assigned. Constraint (3) means that the total processing time of all tasks assigned to a station must not exceed the cycle time. Constraint (4) stands for the precedence constraints. Constraint (5) ensures that, if j−th station is not considered (i. e. , $y_j = 0$), then tasks cannot be assigned to that station (i. e. , $x_{ij}=0$ for all $i \in T_j$). Constraint (6) implies that, if $y_j=0$, then $y_{j+1}=0$ (i. e. , a work station cannot be considered unless a preceding work station is used). Constraint (7) provides a feasible range of cycle time, which is based on a practical range of the desired production rate.

Even though Type Ⅲ ALB formulation results in an optimal mix of the number of work stations and the cycle time, and minimizes the total idle time, it has not been considered in the literature. The reason for this may be the following: (i) the complexity of the integer nonlinear formulation and the lack of efficient algorithms for solving integer nonlinear programming; and (ii) high computer time costs when such algorithms are applied. In general, the type to be chosen may depend on the problem situation. As noted earlier, if the capacity of number of stations is rather inflexible because of fixed size of machines of fixed layout, optimizing means finding the minimum cycle time, given the existing work stations (Type II problem).

If, however, the output of the production line, and consequently, its cycle time, are to be held constant, then the minimum number of stations should be determined (Type I problem). If a new assembly line is designed under flexible conditions, or if frequent rebalancing is necessary as it is under Just−in−time production system, optimal combination of cycle time and the number of work stations may be desired. If multiple products are produced in a batch mode in a flexible assembly system where setup times are low, the Type Ⅲ approach can be applied. In practice, however, all these problems may be subject to another set of constraints (e. g. , budget, floor space, production planning, etc.).

Problem (P−0) can also be formulated and solved as a linear 0−1 integer programming prob-

lem (see Appendix). Since the size of the problem becomes prohibitively large when the number of tasks increases, however, it may not be practically desirable to solve a large scale Type Ⅲ ALB problem (P−0) directly. As an alternative, by solving Type I and Type Ⅱ approaches in combination, much computation time can be saved. The following two methods describe such an approach.

We consider the following methods for finding the optimal mix of c and m by solving 0−1 integer programming problems. Method A utilizes both Type I and Ⅱ approaches. It starts with Type I problem to obtain an optimal number of work stations with a fixed value of the cycle time. The Type Ⅱ problem is then used repeatedly until a global optimal mix of c^* and m^* is found.

Method B is a variant of Method A. It obtains a local optimal mix of c and m.

3.1 Method A

Step 1. (Application of Type I) with $c^0 = c$, solve a revised (P−0) (i. e., (p−1) below) to obtain m^0 and the value of c^0m^0, where m is an initial optimal solution of Type I problem.

Step 2. (Application of Type Ⅱ) Set k = 1.

Step 3. Set $m^k = m^{k-1}$.

If $m^k < \underline{m}$, then go to step 5. otherwise go to step 4.

Step 4. By solving a revised (P−0) (i. e., (P−2) below), obtain c^k and the value of $c^k m^k$. Then set k = k+1 and go to Step 3.

Step 5. Find c^* and m^* such that

$$c^*m^* = \min_{k=1,\cdots,q} \{c^k m^k\}, \text{ where } q=(m^0 - \underline{m}).$$

Problem (P−1)

$$\text{Min} \sum_{j=1}^{\overline{m}} Y_j$$

s.t. constraints (2), (3), (4), (5), (6), (8) and (9).

Problem (P−2)

Min c

$$\text{s.t.} \quad \sum_{i \in T_j} t_i x_{ij} \le c, \qquad j=1,\cdots,m^k \tag{10}$$

$$\sum_{i \in T_j} X_{ij} \le |T_j|, \qquad j=1,\cdots,m^k \tag{11}$$

and constraints (2), (4), (7) and (8).

In Method A, $(m^{\circ} - \underline{m})$ Type II problems and one Type I problem should be solved with a moderate amount of revision and reformulation time. In Step 1, c is replaced with a constant, $\underline{c}$ (or t_{max}, if $\underline{c}$ is not available), which is the lower bound of the feasible range of c, and (P−1) is solved. With a known m^k obtained in Step 3, corresponding Y_j's will be determined and (P−0) will be revised as (P−2). (P−2) is then solved repeatedly with different values of m^k until m^k reaches its lowest limit. Using the value of c^k in Step 4, obtain the optimal solution of (P−1) as the revised value of m^k and compute the revised value of $c^k m^k$. This modification is expected to reduce the total number of (P−1) and (P−2) problems that need to be solved. Since, in general, the range of m is smaller than that of c, fewer integer LP problems will be solved.

In Method B, we compare the values of $c^k m^k$ and repeat while $c^k m^k \geq c^{k+1} m^{k+1}$, stopping when $c^k m^k < c^{k+1} m^{k+1}$. This results in a reduction in number of iterations. However, it does not guarantee an optimal mix of c and m (see Table 2). The modification suggested earlier for the Step 4 of Method A, namely, using c^{k+1} to obtain the optimal solution of (P−1) as the revised value of m^{k+1}, is also applicable in Step 3.

3. 2 Method B

Step 1. (Application of Type I) With $c^{\circ} = \underline{c}$, solve (P−1) and obtain m°.

Step 2. (Application of Type II) Set k=0 and let $Z=c^k m^k$, the current value of c · m at iteration k

Step 3. Set $m^{k+1}=m^k - 1$. If $m^{k+1} < \underline{m}$, then go to Step 5. Otherwise, solve the revised (P−2) with a new m^{k+1} value, and obtain c^{k+1} as an optimal solution and go to step 4.

Step 4. Compare Z with $c^{k+1}m^{k+1}$. If $c^{k+1}m^{k+1} \leq Z$, then stop. Otherwise, set $Z=c^{k+1}m^{k+1}$ and $k=k+1$, and go to Step 3.

Step 5. Stop. c^k and m^k are the best mix of c and m.

4. Computational Results

In order to compare these two methods, we have solved small test problems (Mertens [6], Bowman [7], Jackson [8] and Sawyer [9]).

First, (P−1) is formulated based on the feasible ranges of c and m. The best possible

formulations (i. e., minimum possible number of variables and constraints) were attained by using the methods available in the ALB literature. The problems were then sloved using a computer program written in FORTRAN that has a solution method for solving a zero-one linear program as a subroutine. The elapsed CPU times on a Prime minicomputer and on the Cray Supercomputer X-MP system are obtained. Even though our computer run does not utilize a feasible solution as a starting point, such methods of arbitration could be used to save computation time. Efficient algorithms for Type I and Type II ALB problems which are available in the ALB literature can also be utilized to save computation time. The following tables summarize the comparison between two methods.

〈Table 1〉 Elapsed CPU Time (in seconds)

Problem	Method A		Method B	
	Cray	Prime	Cray	Prime
Mertens (7 task)	2.39	85.96	2.67	79.39
Bowman (8 task)	3.01	71.05	1.36	29.02
Jackson (11 task)	5.60	128.62	3.88	121.23
Sawyer (30 task)	19.45	834.27	16.59	773.61

〈Table 2〉 Optimal Values

Problem	Method A			Method B			Rato(A/B)
	c	m	cm	c	m	cm	
Mertens (7 task)	29	1	29	7	5	35	0.82
Bowman (8 task)	27	3	81	18	5	90	0.90
Jackson (11 task)	12	4	48	7	8	56	0.86
Sawyer (30 task)	41	8	328	35	10	350	0.94

Note : The following arbitrary feasible ranges of c and m were used in the computation.

Problem	c	m
7 task	$6 \leq c \leq 30$	$1 \leq m \leq 7$
8 task	$16 \leq c \leq 32$	$1 \leq m \leq 7$
11 task	$7 \leq c \leq 21$	$1 \leq m \leq 9$
30 task	$34 \leq c \leq 102$	$1 \leq m \leq 10$

formulations (i. e., minimum possible number of variables and constraints) were attained by using the methods available in the ALB literature. The problems were then sloved using a computer program written in FORTRAN that has a solution method for solving a zero-one linear program as a subroutine. The elapsed CPU times on a Prime minicomputer and on the Cray Supercomputer X-MP system are obtained. Even though our computer run does not utilize a feasible solution as a starting point, such methods of arbitration could be used to save computation time. Efficient algorithmes for Type I and Type II ALB problems which are available in the ALB literature can also be utilized to save computation time. The following tables summarize the comparison between two methods.

It can be easily observed that the computation time can be saved by using Method B, even though the optimal solutions cannot be guaranteed. It can also be easily assumed that the number of iterations in Method A and Method B would increase when the ranges of c and m become wider. Extra work for revision and reformulation will also increase in such cases.

5. Conclusions

We have proposed integer programming formulations and solution methods for the classical single model assembly line balancing problem. The new approach not only generalizes the currently existing 0-1 formulations but also guarantees the minimum idle time with the optimal mix of the cycle time and the number of work stations upon solving the problem. This new approach will provide a valuable input for line designers who are often faced with the problem of optimizing line configuration parameters such as the maximization of production rates or the minimization of line installation costs.

Solving Type Ⅲ ALB ((P-0)) directly requires efficient solution algroithms because it may be prohibitively time consuming for practical-sized problems. Either Method A or Method B is suggested as a possible solution method. Even though the nature of Type Ⅲ ALB problem is somewhat complicated in terms of the size and the difficulty of the problem, however, such factors as factory utilization, availability of high speed computer and powerful commercial software packages, and expected savings from line installation costs should be emphasized as favoring the use of Type Ⅲ ALB approaches in same settings.

Future research could be centered on the inclusion of specific cost terms, budgetary restrictions, availability of floor space, capacity constraints as well as the idle time. Development of efficient optimal-seeking algorithms and heuristics which capitalize on special characteristics of Type Ⅲ ALB problems are also desirable.

Appendix

Since the linear 0-1 methods are typically easier than the nonlinear approach, we introduce an approach to transform our nonlinear objective function into a linear relation. Consider the following approach in which every t_i is assumed to be an integer.

Type Ⅲ ALB objective can be further simplified by introducing another variable

$$m \sum_{j=1}^{\overline{m}} Y_j,$$

thereby making the term in objective function:

$$c \cdot m \qquad (A-1)$$

The sum of processing times

$$\sum_{i=1}^{N} t_i,$$

is no longer used because the term is a constant. Taking the log of $(c \cdot m)$, we have the following objective:

$$\text{Minimize } \{\log c + \log m\} \qquad (A-2)$$

Note that we are minimizing the sum of concave functions which is also concave. By also noting that a minimum of a concave function is attained at an extreme point of the feasible re-

gion, we can use a grid linearization.

We choose two sets of grid points $\{c^1,\cdots,c^g\}$ and $\{m^1,\cdots,m^{\bar{m}}\}$.

The set of grid points for c is feasible within its range,

$t_{max} \le c \le \bar{c}$ (if $\bar{c}$ is not available, then $\bar{c} = \sum_{i=1}^{n} t_i$).

The functions log c and log m may be approximated on their sets of grid points by drawing their piecewise linear functions.

The objective function (A-2) is then rewritten as :

$$\text{Minimize } [\min\{l_i(c)\} + \min\{Y_k(m)\}], \tag{A-3}$$

where $l_i(c) = \alpha_i + \beta_i c$ and $Y_k(m) = a_k + b_k m$ are linear functions for log c and log m, respectively and $i=1,\cdots,g$ and $k=1,\cdots,\bar{m}$.

We first replace c as the sum of binary variables,

$$c = t_{max} + \sum_{i=1}^{k} c_i, \tag{A-4}$$

where $k = c - t_{max}$ and c_i is binary. Note that each ci is defined in the specific unit interval within the feasible range of c. log c can then be expressed as:

$$\log c = 1 + \alpha_1 + \beta_1 c_2 + \cdots + \beta_k c_k$$

with the following additional constraints, $c_i \ge c_{i+1}$, for $i=1,\cdots,k-1$ and $\alpha_1 = \log t_{max}$ and $\beta_h = \log h - \log(h-1)$, for $h = t_{max} + 1, t_{max} + 2, \cdots, c$.

Similarly, by using the relationship

$$m = \underline{m} + \sum_{j=\underline{m}+1}^{\bar{m}} Y_j, \tag{A-5}$$

we can rewrite log m as:

$\log m = a_1 + b_1 y_1 + b_2 y_2 + \cdots + b_m y_m$,

where $a_1 = \log \underline{m}$, $b_h = \log (h+1) - \log h$, for $h = \underline{m}, \underline{m}+1,\cdots,\bar{m}-1$.

Here we do not need any additional constraints $Y_j \geq Y_{j+1}$ since we have already included them in our original Type Ⅲ formulation. Our new linear objective function will then become:

$$\text{Minimize } \{(\alpha_1 + \beta_1 c_1 + \cdots + \beta_k c_k) + (a_1 + b_1 y_1 + \cdots + b_{\bar{m}}\, y_{\bar{m}})\} \qquad (A-6)$$

The Type Ⅲ problem (P−0) is formulated as the pure zero−one programming problem:

Minimize (A−6)

s.t. (A−4), (A−5) and constraints (2) − (9)

Even though this approach guarantees global minimum solutions of c and m, the additional binary variables and constraints may be too great if the range of the cycle time is too wide.

References

[1] Kilbridge M. D., and L. Wester. "The balance delay problem," *Mgmt. Sci.* 8, 69−84 (1961).

[2] Baybars, I. "A survey of exact algorithms for the simple assembly line balancing problem," *Mgmt Sci.* 32. 909−932 (1986).

[3] Rosenblatt, J. M. and C. R. Carson. "Designing a production Line to maximize profit," *AIIE Trans.* 3, 117−122 (1985).

[4] Deckro, F. R. "Balancing cycle time and workstations," *AIIE Trans.* 21, 106−111 (1989).

[5] Patterson, J. H. and J. J. Albracht. "Assembly line balancing: 0−1 programming with Fibonacci search," *Opns. Res.* 23, 166−184 (1975).

[6] Mertens, P., "Assembly line balancing by partial enumeration," *Ablauf−und planungforschung* 8, 429−433 (1967).

[7] Bowman, E. H., "Assembly line balancing by linear programming," *Opns. Res.* 8, 385−389 (1960).

[8] Jackson, J. R., "A computing procedure for a line balancing problem," *Mgmt. Sci.* 2, 261−271 (1956).

[9] Sawyer, J. H. F., *Line Balancing*. The Machinery Pub. Co. Sussex, Great Britainw (1970).

[10] Glover, F., "Improved linear integer programming formulations of nonliear integer problems," *Mgmt. Sci.* 22, 455−460 (1975).

[11] Ignall, E. J., "A review of assembly line balancing," *J. of Ind. Eng.* 16, 244−254 (1965).

[12] Thangavelu S. R. and C. M. Shetty, "Assembly line balancing by zero−one integer programming," *AIIE Trans.* 3, 61−68 (1971).