INT. J. PROD. RES., 1990, VOL. 28, NO. 7, 1305–1316

Development of cost model for the single-model stochastic assembly line balancing problem

SUBHASH C. SARIN† and ERDAL EREL‡

In this paper we develop a cost model for the single-model stochastic assembly line balancing problem for the objective of minimizing the total labour cost (dicated by the number of stations on the line) and the expected incompletion cost arising from tasks not completed within the prescribed cycle time. Its use is demonstrated in a dynamic programming procedure which is implemented using a bounding strategy to curtail storage and computational requirements. The solutions obtained are compared with those obtained using the procedure of Kottas and Lau (1973, 1981).

1. Introduction

The single-model, stochastic assembly line balancing problem can be stated as follows: given a finite set of tasks, each having a performance time distributed according to a probability distribution, and a set of precedence relations which specify the permissible orderings of the tasks, the problem is to assign the tasks to an ordered sequence of stations such that the precedence relations are satisfied and some measure of performance is optimized. The problem addressed in this paper has the objective of minimizing the total system cost consisting of the total labour cost and total expected incompletion cost. Total labour cost term is a linear function of the number of stations on the line. Since task performance times are random variables, some tasks cannot be completed within the prespecified cycle time, and these incompletions compose the total expected incompletion cost term. The optimal objective function value is obtained by varying the number of stations and the allocations of tasks to these stations, such that: (1) all tasks are allocated to stations, (2) no task is allocated more than once, (3) if task x precedes task y on the precedence diagram, then y is not allocated to a station that precedes the one to which x is assigned.

Although extensive research has been done on the deterministic version of the problem, relatively less work has been done to develop efficient, optimum-seeking solution procedures for the stochastic version. The problem has a finite but extremely large number of feasible solutions and the problem's inherent integer restrictions result in enormous computational and storage difficulties. The stochasticity of the task performance times are recognized and stated by several authors (Arcus 1966, Freeman and Jucker 1967, Wild 1972). Although some researchers (Kottas and Lau 1973, 1976, 1981, Vrat and Virani 1976, Shtub 1984, Reeve and Thomas 1973, Sculli 1984) have developed procedures (that are basically heuristic) to solve this problem, no one has yet reported an exact development of the cost model. In this paper, we develop such a cost function and demonstrate its implementation in a dynamic programming based scheme.

Revision received March 1989.

† Department of Industrial Engineering and Operations Research, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA.

‡ Department of Management, Hacettepe University and Bilkent University, Ankara, Turkey.

In the sequel, we first present the notation and assumptions in Section 2. The development of the cost model is then given in Section 3. Its implementation in a dynamic programming scheme and results of an experimentation are given in Section 4, and finally some concluding remarks are made in Section 5.

2. Notation and assumptions

The following notation will be used throughout the paper.

- C cycle time
- N number of tasks in the problem
- IC_i incompletion cost of task i, for $i=1,\ldots,N$
- K number of stations on the line
- L labour rate
- t_i performance time of task i, for $i=1,\ldots,N$
- W set of all the tasks in the problem

The formulation of the problem is developed based on the following assumptions:

(1) Task performance times are random variables. They are independent of each other and the parameters of the distributions are known.
(2) No splitting of the tasks among stations is permitted.
(3) Each station is manned by one worker with uniform wage.
(4) Demand rate is deterministic.
(5) No buffer inventory between the stations is allowed.
(6) The tasks assigned to a station are performed in a given order.
(7) Whenever a task is not finished, the unit moves down the line with as many of the remaining tasks being completed as possible.
(8) Incomplete tasks are completed off the line at a cost which is not dependent on the fraction of the task completed on the line.
(9) No blocking due to incomplete tasks.

Assumption 1 is a direct consequence of the fact that most of the tasks are manually performed and variances in the performance times of these tasks are inevitable. Normal distribution is the most frequently assumed distribution for the performance times in the literature (Chakravarty and Shtub 1986, Kottas and Lau 1973, 1976, 1981, Vrat and Virani 1976). If the assumed probability distribution can take negative values, then it should be truncated at zero. On the other hand, the truncation at zero can be made if the probability that the random variable can take negative values is small enough. Suppose that the performance times are independent, normally distributed random variables, and let μ_i and σ_i^2 be the mean and variance of the performance time of task i. If we let E represent the area to the left of zero under a normal distribution, ε a small quantity greater than zero, $\Phi(\cdot)$ be the cumulative standard normal distribution function and $\sigma_i = a \times \mu_i$ for all i, where a is a constant, then the area to the left of zero can be ignored if

$$E=\Phi\left(\frac{-\mu_i}{\sigma_i}\right)\leqslant\varepsilon \quad \text{or} \quad a\leqslant -\frac{1}{\Phi^{-1}(\varepsilon)} \quad \text{for } i=1,\ldots,N.$$

As is shown by Erel (1987), for small ε, the upper bound values of a are of realistic magnitudes from a practical point of view. In addition, it is also shown that when the performance times are assumed to be normally distributed, the effect of truncation at zero is negligible for tasks of realistic assembly lines (Erel 1987, Wilhelm 1987).

Assumptions (2)–(5) are similar to those made in the majority of the line balancing literature. However, the formulation presented in this paper could be easily extended to relax assumption (3); more than one worker could be assigned to a station, nonidentical wage rates could be applied for different tasks, and other constraints, such as zoning or positional constraints, could be imposed on making station assignments. Assumptions (6) and (7) represent only one of the ways the incompletion situations are handled; the formulation could be modified to handle other incompletion situations, such as the incompleted units are scrapped, or additional workers are hired that can help the workers at stations encountering incompletions.

Incompletion cost is the cost of completing the task off the line and is calculated as if the incomplete task is handled by a group of workers supporting the line. Incompletion cost of task i, IC_i is assumed to be larger than $L \times \mu_i$ for $i = 1, \ldots, N$. A task becomes incomplete due to two reasons: (*a*) the task is not completed within the cycle time, or (*b*) the task is a follower of another incomplete task on the precedence diagram. When a task is incomplete, a set of tasks may not get started to be processed. This set of tasks depends on the precedence diagram and the allocations of tasks to stations.

The demand rate for the product is assumed to be known with certainty. A fixed demand rate imposes a fixed cycle time. The model will be developed to give a solution for the cycle time imposed by the demand rate.

3. Development of the cost model

The objective function of the single-model, stochastic assembly line balancing problem stated in Section 1 is as follows:

$$\text{Min } Z = \text{Total labour cost} + \text{total expected incompletion cost.}$$

In this section, a general expression will be developed that captures the cost terms of the above objective function for a given number of stations and allocations of tasks to these stations. As alluded to earlier, the optimal solution can then be obtained by varying the number of stations and the allocations of tasks to these stations. To that end, we first introduce some additional notation.

Let the set of tasks following task i on the precedence diagram and in the station which contains task i be denoted by A_i' and H_i, respectively. Note that when task i is not completed within C, then the tasks in H_i cannot be started. Moreover, the tasks in

$$\bigcup_{j \in H_i} A_j'$$

also cannot be started. Let

$$A_i = A_i' \bigcup H_i \bigcup \left(\bigcup_{j \in H_i} A_j' \right)$$

Hence, incompletion of task i incurs a cost equal to the incompletion cost of task i and that of the tasks in A_i; that is

$$\sum_{j \in A_i} IC_j + IC_i$$

Let P_i and B_i be the set of tasks preceding task i on the precedence diagram and in the station which contains task i, respectively. Task i can be started only if the tasks in P_i and the tasks in B_i, that can be started are completed. Let $j \in B_i$ and $j \notin P_i$, and if j is started to be processed, then it should be completed within C for task i to get started,

although $j \notin P_i$. Note that if there are n tasks in B_i that do not belong to P_i, then there can be at most 2^n starting events for task i. Let T_i be the set of tasks in B_i that do not belong to P_i. Then, $T_i = B_i \cap (W - P_i)$. Let T_i^j be the jth starting event of task i in which the tasks in TS_i^j and task i can be started and the tasks in TN_i^j cannot be started. Note that T_i^j does not imply that job i will necessarily be started. Of course, job i can be started only if jobs in TS_i^j are completed. We address these possibilities below. Moreover, note that

$$\{B_i \cap P_i\} \subseteq \mathrm{TS}_i^j \quad \text{and} \quad B_i = \mathrm{TS}_i^j \cup \mathrm{TN}_i^j \quad \text{for } j = 1, \ldots, 2^n.$$

In order to compute the cost terms of the objective function, several variables should be determined. These variables include the probability that a task can get started to be processed, the probability that a task is not completed within C after it has been started, and the cost incurred due to the incompletion of a task. As some tasks are common followers of other tasks that are in parallel on the precedence diagram, their contributions in the total cost expression need to be appropriately handled to avoid overcounting. Next, we derive the probability that a task is not completed within C after it has been started to be processed.

Let γ_i^j denote the probability that T_i^j occurs, and β_i^j denote the probability that T_i^j occurs and task i is not completed within C while the tasks in TS_i^j are completed within C. Then, β_i^j can be expressed as follows:

$$\beta_i^j = \gamma_i^j \times [\Pr(\text{task } i \text{ incomplete and tasks in } \mathrm{TS}_i^j \text{ complete, given } T_i^j)]. \tag{1}$$

To compute Pr(task i incomplete and tasks in TS_i^j complete), let X and Y be the events representing that the task i is not completed within C and tasks in TS_i^j are completed within C, respectively. (Accordingly, $\bar{Y}$ represents the event that the tasks in TS_i^j are not completed within C.) Then, $\Pr\{X \text{ and } Y\} = \Pr\{X/Y\} \cdot \Pr\{Y\}$. Note that

$$\Pr\{X/Y\} = \frac{\Pr\{X\} - \Pr\{X/\bar{Y}\}\Pr\{\bar{Y}\}}{\Pr\{Y\}}$$

Note also that $\Pr\{X/\bar{Y}\} = 1$, because this represents the probability that task i is incomplete given that the tasks in TS_i^j are incomplete (and the tasks in TS_i^j precede task i in the station). Hence, $\Pr\{X \text{ and } Y\} = \Pr\{X\} - \Pr\{\bar{Y}\}$. Then, expression (1) can be rewritten as follows:

$$\begin{aligned}\beta_i^j &= \gamma_i^j \times [\Pr(\text{task } i \text{ incomplete}) - \Pr(\text{tasks in } \mathrm{TS}_i^j \text{ incomplete})] \\ &= \gamma_i^j \times \left\{ \Pr\left[\sum_{k \in \mathrm{TS}_i^j} t_k + t_i > C \right] - \Pr\left[\sum_{k \in \mathrm{TS}_i^j} t_k > C \right] \right\} = \gamma_i^j \times \Gamma_i^j \end{aligned} \tag{2}$$

where,

$$\Gamma_i^j = \Pr\left[\sum_{k \in \mathrm{TS}_i^j} t_k + t_i > C \right] - \Pr\left[\sum_{k \in \mathrm{TS}_i^j} t_k > C \right]. \tag{3}$$

If we assume that task performance times are normally distributed random variables with known means and variances, and since they are independent of each other and of the ordering in stations, then (3) becomes

$$\Gamma_i^j = \left[1 - \Phi\left(\frac{C - \left[\sum_{k \in \mathrm{TS}_i^j} \mu_k + \mu_i \right]}{\left[\sum_{k \in \mathrm{TS}_i^j} \sigma_k^2 + \sigma_i^2 \right]^{1/2}} \right) \right] - \left[1 - \Phi\left(\frac{C - \left[\sum_{k \in \mathrm{TS}_i^j} \mu_k \right]}{\left[\sum_{k \in \mathrm{TS}_i^j} \sigma_k^2 \right]^{1/2}} \right) \right]. \tag{4}$$

Let β_i denote the probability that task i is not completed within C after it has been started to be processed. If there are n tasks in T_i, then, as discussed before, there can be at most 2^n different starting events of task i, and β_i can be expressed as follows:

$$\beta_i = \sum_{j=1}^{2^n} \beta_i^j = \sum_{j=1}^{2^n} \gamma_i^j \times \Gamma_i^j. \tag{5}$$

Consider tasks x and y such that $x \in \mathrm{TS}_i^j$ and $y \in \mathrm{TN}_i^j$. For T_i^j to occur, at least one task in P_y should not be completed within C. If $P_y \subseteq P_x$, then T_i^j is an infeasible starting event, since all the tasks in P_x should be completed within C. Hence, let $fs_i (\leqslant 2^n)$ denote the number of feasible starting events for task i. In reality, fs_i is much smaller than 2^n. Consequently,

$$\beta_i = \sum_{j=1}^{fs_i} \beta_i^j. \tag{6}$$

In the computation of β_i, the determination of Γ_i^j is straightforward once the assignment of tasks to a station is known. However, the determination of γ_i^j is not as straightforward because of the complexity of its occurrence. Here, we discuss a procedure to compute γ_i^j that uses a special enumeration tree in which all possible ways of realizing the starting event T_i^j are represented, and the occurrence probabilities of these cases are computed. Consequently, the occurrence probability of the starting event T_i^j, γ_i^j is derived. Let $w_{i,k}^j$ denote the probability of occurrence of the kth case in the enumeration tree constructed to compute γ_i^j, and let $\mathrm{IW}_{i,k}^j$ and $\mathrm{CW}_{i,k}^j$ denote the sets of tasks that are incomplete and and complete in the kth case of the tree, respectively. Let l_i denote the station to which task i is assigned and q_{b_i} denote the number of tasks assigned to station b_i. The tree has $l_i - 1$ levels and each level has q_{b_i} sublevels for $b_i = 1, \ldots, l_i - 1$. The levels and sublevels represent the stations and the tasks assigned to the stations, respectively. To illustrate this procedure consider the precedence diagram of an 11-task problem and an allocation of its tasks to stations as depicted in Fig. 1. The tree constructed to compute γ_8^j, where $\mathrm{TS}_8^j = \{5, 6\}$ and $\mathrm{TN}_8^j = \{7\}$, is depicted in Fig. 2. Note that for starting event T_8^j to occur, one or more tasks in P_7 should not be completed, and all the tasks in P_5, P_6 and P_8 should be completed. Note also that $P_5 = \{1\}$, $P_6 = \{1, 2\}$, $P_7 = \{1, 3, 4, 5\}$ and $P_8 = \{1, 2, 6\}$.

The numbers outside the nodes are the node numbers and the ones inside the nodes represent the tasks in the nodes. Nodes with tasks i and $\bar{i}$ represent whether task i is completed or not completed within C, respectively. Level 1 corresponds to station 1 and the first sublevel of level 1 represents task 1. Task 1 is either completed or not completed within C, and these events are represented by nodes 1 and 2. If task 1 is completed, then task 3 can be started, and task 3 is either completed or not completed

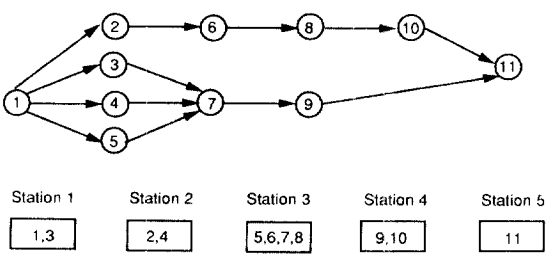

Figure 1. Precedence diagram of the example problem and an allocation of its tasks to stations.

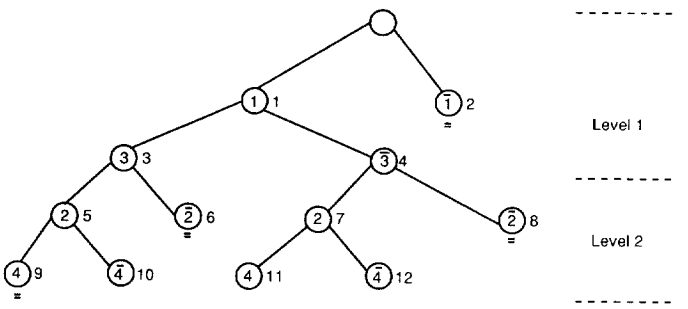

Figure 2. Probability enumeration tree to compute γ_8^j of the example.

within C, and is represented by nodes 3 and 4. Note that for starting event T_8^j to occur, task 1 has to be completed, since $1 \in P_5, P_6, P_8$; thus, node 2 is pruned. Level 2 represents the second station. Task 2 can be started whether task 3 is completed or not, since $3 \notin P_2$. Task 2 is either completed or not completed within C, as represented by nodes 5–8. If task 2 is completed, then task 4 can be started. On the other hand, task 2 has to be completed for starting event T_8^j to occur, since $2 \in P_6, P_8$. Thus, nodes 6 and 8 are pruned and not branched into descendent nodes. Nodes 5 and 7 are branched into nodes representing task 4 being completed or not completed within C. Note that for starting event T_8^j *to* occur, one or more tasks in P_7 has to be incomplete; thus, node 9 represents an infeasible case, since none of the tasks in P_7 are incomplete in the case represented by node 9. The cases represented by nodes 10, 11 and 12 are the all possible cases for T_8^j to occur. The tasks that are complete and incomplete in these three cases are given below:

$$\mathrm{CW}_{8,1}^j = \{1,2,3\} \quad \text{and} \quad \mathrm{IW}_{8,1}^j = \{4\} \text{ corresponding to node 10}$$

$$\mathrm{CW}_{8,2}^j = \{1,2,4\} \quad \text{and} \quad \mathrm{IW}_{8,2}^j = \{3\} \text{ corresponding to node 11}$$

$$\mathrm{CW}_{8,3}^j = \{1,2\} \quad \text{and} \quad \mathrm{IW}_{8,3}^j = \{3,4\} \text{ corresponding to node 12}$$

In the enumeration tree discussed above, a node at sublevel v with task j is branched into descendent nodes at sublevel $v+1$ with task i assigned to them if all the tasks in P_i are completed in the case represented by the parent node at sublevel v. Otherwise, if the parent node represents a case in which one or more tasks in P_i are incomplete, then the node is not branched into nodes at sublevel $v+1$; sublevel $v+1$ is skipped. Within a level, if the parent node represents a task being not completed, and if the node can be branched into nodes of the next sublevel, then the parent node is branched into a descendent node representing the task being incomplete. A node is pruned if the task represented by the node is incomplete, and that task is required to be completed for the associated starting event to occur.

Next, we discuss the computation of the occurrence probabilities of the nodes in the enumeration tree. Let O_u and $O_{\bar{u}}$ represent the occurrence probabilities of a pair of descendent nodes with task u assigned. Note that this pair of nodes are branched from a common parent node and let o_i denote the occurrence probability of the parent node. Then, $o_i = o_u + o_{\bar{u}}$. The occurrence probabilities of the nodes are computed in a similar manner as the computation of Γ_i^j. If the performance times are normal random

variables, then the occurrence probabilities of the nodes in the enumeration tree (depicted in Fig. 2) are computed as follows:

$$o_1 = \Phi\left(\frac{C-\mu_1}{\sigma_1}\right) \quad \text{and} \quad o_2 = 1 - o_1$$

$$o_3 = \Phi\left(\frac{C-(\mu_1+\mu_3)}{(\sigma_1^2+\sigma_3^2)^{1/2}}\right) \quad \text{and} \quad O_4 = o_1 - o_3$$

$$o_5 = o^3 \times \Phi\left(\frac{C-\mu_2}{\sigma_2}\right) \quad \text{and} \quad o_6 = o_3 - o_5$$

$$o_7 = o_4 \times \Phi\left(\frac{C-\mu_2}{\sigma_2}\right) \quad \text{and} \quad O_8 = o_4 - o_7$$

$$o_9 = o_3 \times \Phi\left(\frac{C-(\mu_2+\mu_4)}{(\sigma_2^2+\sigma_4^2)^{1/2}}\right) \quad \text{and} \quad o_{10} = o_5 - o_9$$

$$o_{11} = o_4 \times \Phi\left(\frac{C-(\mu_2+\mu_4)}{(\sigma_2^2+\sigma_4^2)^{1/2}}\right) \quad \text{and} \quad o_{12} = o_7 - o_{11}$$

Now we can express the occurrence probability of starting event T_8^j, γ_8^j as the summation of the occurrence probabilities of nodes 10, 11 and 12. That is,

$$\gamma_8^j = o_{10} + o_{11} + o_{12} = w_{8,1}^j + w_{8,2}^j + w_{8,3}^j$$

In general, if there are fw_i^j cases in the tree for the starting event T_i^j to occur, then the occurrence probability of the starting event T_i^j, γ_i^j can be expressed as follows:

$$\gamma_i^j = \sum_{k=1}^{fw_i^j} w_{i,k}^j.$$

With the determination of the probability of occurrences of desired events, the contribution of task i to the total expected incompletion cost can now be determined as follows. First, we illustrate this for an example. Consider again task 4 in station 2 of the example above. There is only one starting event for task 4 with $TS_4^1 = \{2\}$ and $TN_4^1 = 0$. In the probability enumeration tree constructed for this starting event, there are two cases corresponding to task 3 being completed or not. Let the probability of the case in which task 3 is not completed be denoted by $w_{4,1}^1$. When this case occurs, an expected incompletion cost of $w_{4,1}^1$ $(IC_3 + IC_7 + IC_9 + IC_{11})$ is incurred. Note that $A_3 = \{7, 9, 11\}$. The probability that this case of starting event T_4^1 occurs and task 4 is not completed within C is $w_{4,1}^1 \times \Gamma_4^1$. The incompletion cost of the tasks in $A_4 = \{7, 9, 11\}$ are multiplied with this probability and added to the total expected incompletion cost term. Note that the incompletion cost of tasks 7, 9 and 11 is overcounted with probability $(w_{4,1}^1 \times \Gamma_4^2)$, and should be subtracted from the total system cost expression. In general, the set of tasks whose incompletion costs are overcounted corresponding to the kth case of the starting event T_i^j is

$$CF_{i,k}^j = A_i \cap \left(\bigcup_{m \in IW_{i,k}^j} A_m \right).$$

Note that the incompletion costs of the tasks in $CF_{i,k}^j$ are overcounted with probability $(w_{i,k}^j \times \Gamma_i^j)$. If SB_i^j denotes the overcounted incompletion costs corresponding to T_i^j, then

$$SB_i^j = \sum_{k=1}^{fw_i^j} \Gamma_i^j \times w_{i,k}^j \times \left[\sum_{m \in CF_{i,k}^j} IC_m \right]. \tag{8}$$

Now, the expression for the contribution of task i to the total expected incompletion cost term can be defined. When task i is started and not completed within C, then this event incurs a cost of

$$\left[IC_i + \sum_{j \in A_i} IC_j \right]$$

and the expected incompletion cost of task i can be expressed as

$$\beta_i x \left(IC_i + \sum_{j \in A_i} IC_j \right).$$

The computation of the total labour cost term of the objective function for a given number of stations (K) is straightforward. The total labour cost term is linearly proportional to the number of stations on the line; the proportionality constant is CL. On the other hand, the total expected incompletion cost term is a monotonically nonincreasing function of K. As K increases, the total expected incompletion cost term decreases to an asymptote. Note that even with the maximum number of stations ($K = N$), there may remain a positive total expected incompletion cost; this quantity constitutes the asymptote.

The cost factors of the objectives function stated above can now be generalized to represent the total system cost function of a given allocation of tasks to a given number of stations as follows:

$$Z = C \times K \times L + \sum_{i=1}^{N} \left\{ \beta_i \left[IC_i + \sum_{k \in A_i} IC_k \right] - \sum_{j=1}^{fs_i} SB_i^j \right\} \tag{9}$$

The objective is to minimize Z by varying K and the allocations of tasks to these stations.

4. Implementation in a dynamic programming scheme

In the dynamic programming formulation of the problem, the stages are the stations on the line. The state variable at stage n, s_n represents the set of tasks available for assignment at that stage. The decision variable at stage n, $x_n \in X_n$ represents the set of tasks to be assigned to station n, where X_n is the set of all possible sequences of tasks that can be assigned at stage n given s_{n+1}. The return function at stage n, $r_n(x_n, s_{n+1})$ is the total expected cost corresponding to decision variable x_n, and state variable s_{n+1}. In determining the decision variables, the precedence constraints are the only restrictions considered.

The return function, $r_n(x_n, s_{n+1})$ is similar to the objective function developed above and is as follows:

$$r_n(x_n, s_{n+1}) = C \times L + \sum_{i \in x_n} \left\{ \beta_i \left[IC_i + \sum_{k \in A_i} IC_k \right] - \sum_{j=1}^{fs_i} SB_i^j \right\}. \tag{10}$$

If $f_n^*(s_{n+1})$ represents the cost of assigning the tasks in the set $\{s_1 - s_{n+1}\}$ to stages 1 to n, then the recursive relationship can be represented as:

$$f_n^*(s_{n+1}) = \min_{x_n \in X_n} \{r_n(x_n, s_{n+1}) + f_{n-1}^*)s_n)\} \quad \text{for } n = 1, \ldots, N \tag{11}$$

where $s_n = s_{n+1} + x_n$ and $f_0^*(\cdot) = 0$ and $s_{N+1} = 0$.

One of the state variables is the null state at each stage except for the first one; the null state indicates the assignment of all the tasks. The associated f_n^* function value to the null state, $f_n^*(s_{n+1} = 0)$ gives the optimal solution of the problem with n stations.

This dynamic programming formulation of the problem would only obtain the solutions of problems of limited size, because of the excessive number of state and decision variables generated at each stage. Thus, we implement it in a heuristic manner by pruning decision variables that are not expected to lead to the optimal solution. A sufficient number of decision variables at each stage should be pruned, so that the problem could be solved on the computer in a reasonable amount of time. On the other hand, the pruning of the decision variables should not result in a design with an operating cost much higher than the optimal design cost.

We will call the strategy that prunes some of the decision variables at each stage as the 'bounding strategy'. The bounding strategy imposes an upper bound on the incompletion probability of the decision variables. In other words, decision variables that have incompletion probabilities larger than bound provided by the user are pruned. If α denotes this bound, then the decision variable, x_n is pruned if:

$$1 - \Phi\left(\frac{C - \sum_{i \in x_n} \mu_i}{\left(\sum_{i \in x_n} \sigma_i^2\right)^{1/2}}\right) > \alpha. \tag{12}$$

Theoretically, the range of α is between zero and one. The value of one corresponds to the generation of all possible decision variables at each stage. Since it can be assumed that $\Phi(x) = 0{\cdot}0$ for $x \leqslant -3{\cdot}0$, a decision variable is assumed to be incomplete with certainty if the following condition is met:

$$\frac{C - \sum_{i \in A} \mu_i}{\left(\sum_{i \in A} \sigma_i^2\right)^{1/2}} \leqslant -3{\cdot}0. \tag{13}$$

As α is decreased, the decision variables that have incompletion probabilities greater than α are discarded; this process decreases the computational and storage requirements of the dynamic programming formulation. However, the probability of missing the optimal design increases as α is decreased.

The performance of the procedure was compared with the performance of the technique of Kottas and Lau (1973, 1981) on some randomly generated problems. The technique of Kottas and Lau (1973) is a single-pass technique; in other words, once a decision is made to assign a task to a station, the task is never considered again, although further improvement could be made by reconsidering the task for assigning to a different station. The technique could be summarized as follows: an available list is formed by identifying the tasks with no unassigned predecessors. The list is updated

each time a task is assigned. Then, a desirable list is formed by identifying the available list tasks which are marginally desirable for assignment. A task is considered marginally desirable when its anticipated labour savings in the specific position under consideration is larger than its expected incompletion cost. The tasks with virtual certainty of completion are assigned first in descending order of their incompletion costs. These tasks compose the sure list. If the sure list is empty, then the desirable list tasks are assigned in the ascending order of their incompletion costs. When the desirable list gets empty, a new station is established. The tasks which are never marginally desirable are assigned as the first tasks in stations as soon as they are placed in the available list. The procedure continues until the available list gets empty. The procedure is computationally very attractive; up to 50-task problems have been reported to be solved in under 0·1 seconds CPU time on an IBM 360/75. Kottas and Lau (1981) refined their technique to generate several promising line designs. This approach is conceptually related to the techniques of Arcus (1966) and Tonge (1965). Several line designs are generated by utilizing various selection rules for the desirable list tasks. Dominated designs are eliminated and the remaining ones are evaluated; the design with the lowest cost constitutes the solution.

Example no.	No. of tasks	F-ratio	C†	RAN_1	RAN_2
1	11	0·000	58·4	0·052	0·083
2	11	0·000	72·2	0·043	0·076
3	11	0·000	99·9	0·046	0·062
4	11	0·491	23·8	0·057	0·056
5	11	0·491	37·6	0·048	0·099
6	11	0·491	51·4	0·059	0·092
7	11	0·418	65·3	0·051	0·085
8	11	0·418	79·1	0·042	0·079
9	11	0·418	93·0	0·053	0·072
10	11	0·800	57·9	0·052	0·068
11	11	0·800	92·4	0·053	0·058
12	11	0·800	50·8	0·045	0·090
13	15	0·000	99·2	0·057	0·073
14	15	0·000	57·6	0·050	0·055
15	15	0·000	16·0	0·042	0·088
16	15	0·257	64·4	0·054	0·070
17	15	0·257	22·8	0·046	0·053
18	15	0·257	71·2	0·058	0·085
19	15	0·781	29·6	0·050	0·068
20	15	0·781	78·0	0·042	0·050
21	15	0·781	15·6	0·050	0·099
22	16	0·575	84·8	0·046	0·065
23	16	0·575	43·2	0·058	0·098
24	16	0·575	91·6	0·051	0·080
25	17	0·382	50·0	0·043	0·063
26	17	0·382	98·4	0·055	0·095
27	17	0·382	56·8	0·047	0·078
28	18	0·379	15·2	0·059	0·060
29	18	0·379	63·6	0·051	0·093
30	18	0·379	22·0	0·043	0·075

† Cycle times are in minutes.

Table 1. Parameters of the example problems solved.

Example no.	Kottas and Lau solution	DP procedure solution	% difference between the procedures (KL − DP)	DP procedure requirements		
				Number of decision variables	Number of state variables	CPU time†
1	34·420	34·420	0·0	30	27	0·44
2	25·372	25·372	0·0	50	33	0·48
3	35·842	35·342	1·4	61	37	0·49
4	9·880	8·833	10·6	271	166	1·63
5	11·810	9·930	15·9	314	244	2·41
6	15·576	15·487	0·6	284	237	2·13
7	41·202	37·938	7·9	153	138	0·94
8	50·425	44·041	12·7	123	111	0·74
9	42·873	42·873	0·0	94	133	0·73
10	25·346	22·446	11·4	397	688	9·84
11	33·264	32·595	2·0	447	734	11·05
12	19·946	19·937	0·0	391	747	11·53
13	62·104	61·753	0·6	91	57	0·70
14	28·828	28·806	0·1	94	61	0·71
15	10·064	10·063	0·0	74	51	0·67
16	25·799	25·766	0·1	359	192	2·01
17	11·434	11·430	0·0	167	166	1·02
18	36·375	32·107	11·7	252	170	1·40
19	15·462	15·880	−2·7	598	1751	47·07
20	37·575	33·811	10·0	1504	2076	112·20
21	8·831	8·644	2·1	524	1544	38·30
22	33·007	29·038	12·0	2788	1269	103·69
23	22·537	19·348	14·2	1246	991	26·88
24	52·219	52·219	0·0	1513	1151	36·23
25	31·654	29·114	8·0	770	553	9·73
26	75·012	71·919	4·1	862	550	10·74
27	28·734	25·928	9·8	810	584	10·99
28	8·580	8·563	0·2	901	741	11·63
29	30·085	27·246	9·4	1175	737	13·15
30	14·851	13·688	7·8	568	568	5·66

† CPU times are in seconds on an IBM 3090.

Table 2. Solutions of the example problems and the DP procedure (with $\alpha = 0{\cdot}5$) storage and computational requirements.

The problems generated have 18 or fewer tasks and the following parameters: $\mu_i \sim U[0, C]$ with $\sigma_i = \mathrm{RAN}_1$ (μ_1), $\mathrm{IC}_i = \mathrm{RAN}_2$ (μ_i) and $L = \$3{\cdot}00$/hour. RAN_1 and RAN_2 are the multipliers used to obtain the variance and the incompletion costs of the tasks, respectively. Number of tasks, F-ratio, C, RAN_1, and RAN_2 values of the problems are depicted in Table 1. Note that F-ratio (flexibility ratio) is a measure of the number of feasible sequences that can be generated; it ranges from zero for precedence diagrams ordered serially to one for diagrams having no precedence relationships.

Table 2 depicts the solution values of the example problems obtained with the technique of Kottas and Lau and the DP procedure with $\alpha = 0{\cdot}5$. Note that 0·5 is a lower bound on α, since $\mu_i \sim U[0, C]$ for all i. The performance of the DP procedure depends on the value of α, the larger the value of α, the better is the performance. Table 2 also depicts the storage and computational requirements of the DP procedure. The performance of the technique of Kottas and Lau is affected due to the fact that tasks are never reconsidered after being assigned to stations. Another reason is that the marginal

desirability of a task is determined only by examining the expected performance time, incompletion cost, and the position of the task in the station. The probability that a task under consideration is started should also be a factor in determining the marginal desirability of the task. Only in one of the 30 problems (namely, Problem 19), the DP procedure with $\alpha = 0{\cdot}5$ performed worse than the technique of Kottas and Lau. On average, the DP procedure results in solutions with values 5·0% lower than those of the Kottas and Lau technique; 90% confidence interval on this value are 3·3% and 6·7%, respectively.

5. Concluding remarks

In this paper we have developed a cost model for the single-model, stochastic assembly line balancing problem. Even though this problem for the objective relating to the minimization of labour cost and incompletion cost has been addressed by others in the literature, yet no closed form cost function has been reported. This cost function is implemented in a dynamic programming scheme. A heuristic implementation of this procedure is shown to generate solutions that are about 5% better than those generated by the technique of Kottas and Lau.

For problems with larger number of tasks, we have developed an approximation procedure which divides the problem into subproblems. For each subproblem, an approximate solution is obtained using the DP procedure and the cost model presented in this paper. These approximate solutions are further improved by a branch-and-bound type of procedure. The improved solutions of the subproblems are then appended to each other to produce the solution of the original problem. The details of this study will be reported in a forthcoming paper.

References

ARCUS, A. L., 1966, COMSOAL: A computer method of sequencing operations for assembly lines. *International Journal of Production Research*, **4**, 259–277.

CHAKRAVARTY, A. K., and SHTUB, A. 1986, A cost minimization procedure for mixed model production lines with normally distributed task times. *European Journal of Operational Research*, **23**, 25–36.

EREL, E., 1987, A methodology to solve single-model, stochastic assembly line balancing problem and its extensions. Unpublished doctoral dissertation.

FREEMAN, D. R., and JUCKER, J. V., 1967, The line balancing problem. *The Journal of Industrial Engineering*, **18**, 361–364.

KOTTAS, J. F., and LAU, H. S., 1973, A cost oriented approach to stochastic line balancing, *AIIE Transactions*, **5**, 164–171.

KOTTAS, J. F., and LAU, H. S., 1976, A total operating cost model for paced lines with stochastic task times. *AIIE Transactions*, **8**, 234–240.

KOTTAS, J. F., and LAU, H. S., 1981, A stochastic line balancing procedure. *International Journal of Production Research*, **19**, 177–193.

SCULLI, D., 1984, Short term adjustments to production lines. *Computers and Industrial Engineering*, **8**, 53–63.

SHTUB, A., 1984, The effect of incompletion cost on the line balancing with multiple manning of work stations. *International Journal of Production Research*, **22**, 235–245.

REEVE, N. R., and THOMAS, W. H., 1973, Balancing stochastic assembly lines. *AIIE Transactions*, **5**, 223–229.

TONGE, F. M., 1965, Assembly line balancing with probabilistic combinations of heuristics. *Management Science*, **11**, 727–735.

VRAT, P., and VRANI, A., 1976, A cost model for optimal mix of balanced stochastic assembly line and the modular assembly system for a customer oriented production system. *International Journal of Production Research*, **14**, 445–463.

WILD, R., 1972, *Mass-production Management* (New York: John Wiley).

WILHELM, W. E., 1987, On the normality of operation times in small-dot assembly systems: a technical note. *International Journal of Production Research*, **25**, 145–149.