omega

www.elsevier.com/locate/orms

A methodology for solving single-model, stochastic assembly line balancing problem

Subhash C. Sarin[a,*], Erdal Erel[b], Ezey M. Dar-El[c]

[a]*Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*
[b]*Faculty of Business Administration, Bilkent University, 06533 Bilkent, Ankara, Turkey*
[c]*Department of Industrial Engineering and Management, Technion–Israel Institute of Technology, Haifa 32000, Israel*

Received 1 June 1992; accepted 1 January 1999

Abstract

In this paper, a methodology is developed to solve the single-model, stochastic assembly line balancing problem for the objective of minimizing the total labor cost and the expected incompletion cost arising from tasks not completed within the prescribed cycle time. The methodology is based on determining an initial DP based solution and its improvement using a branch-and-bound procedure which uses an approximate solution instead of a lower bound for fathoming nodes. Detailed experimentation shows the superiority of this method over the most promising one from the literature. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Assembly; Line balancing; Heuristics

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 with a prespecified cycle time such that the precedence relations are satisfied and some performance measure is optimized. The problem addressed in this paper has the objective of minimizing the total system cost comprising of the total labor cost and the total expected incompletion cost. Since task performance times are random variables, some tasks cannot be completed within the operating cycle time; these products consequently move down the line with as many of the remaining tasks being completed as possible. Incomplete tasks are finally completed off the line. This incompleted work constitutes the incompletion cost. Such situations of completing the incomplete work off-line is encountered in the assembly of automobiles and appliances.

Although extensive research has been reported in the literature on the deterministic version of the problem, relatively less work has been done to develop efficient, optimum-seeking solution procedures for the stochastic case. The problem has a finite but extremely large number of feasible solutions and the inherent integer restrictions result in enormous computational and storage difficulties. The stochasticity of the task performance times has been recognized and stated by several authors [1,4–19]. Methods for this stochastic line balancing problem that are extensions of the procedures developed for the deterministic version of this problem are reported in [1,6,10,12,18]. Kottas and Lau [7–9]

* Corresponding author. Tel.: +1-540-231-6656; fax: +1-540-231-3322.

0305-0483/99/$ - see front matter
PII: S0305-0483(99)00016-X

developed heuristic procedures for this problem which try to balance the labor and incompletion costs, while Vrat and Virani [19] and Shtub [16] present procedures that are basically similar to the method of Kottas and Lau [7]. A practical procedure to balance stochastic lines is reported by Sculli [15]. Silverman and Carter [17] and Lau and Shtub [10] present a procedure for the case when the assembly line is stopped to complete the incompleted work, while Reeve and Thomas [13] present a comparison of the procedures based on trade and transfer, branch and bound, heuristic branch and bound and a procedure based on combination of branch and bound and trade and transfer ideas.

In this paper, we develop a heuristic enumeration method for the stochastic assembly line balancing problem. Our results are then compared with those obtained by Kottas and Lau [7] whose method is reported as one of the effective methods for solving the problem. It is important in this connection to note at the outset that our method employs one assumption (assumption 4 below) not made by Kottas and Lau [7], and the comparative results relate only to circumstances where this assumption holds.

We first present the notation and assumptions in Section 2; the solution methodology is described in Section 3; the detailed experimentation in Section 4 and finally, conclusions are given in Section 5.

2. Notation and assumptions

The following notation is used:

C = cycle time
IC_i = incompletion cost of task i, for $i = 1, \ldots, N$ (dollar/task)
K = number of stations on the line
L = labor rate (US$/unit time)
t_i = performance time of task i which is a random variable with mean μ_i and variance σ_i^2
W = set of all the tasks assigned to the assembly line

Before proceeding with the analysis of the problem, the following assumptions are made.

1. Whenever a task is not finished, the unit moves down the line with as many of the tasks that do not violate the precedence relations being completed as possible.
2. Incomplete tasks are completed off the line at a cost which is not dependent on the fraction of the task completed on the line.
3. No blocking occurs due to incomplete tasks.
4. Task performance times are independently distributed random variables whose distributions are characterized by a single parameter, implying that μ_i and σ_i are related.
5. No splitting of tasks is permitted among stations.
6. All line workers are paid the same hourly rate and each station is manned by one worker.
7. Demand rate is deterministic, or alternatively cycle time is fixed.
8. No buffer inventory is allowed between stations.
9. 'Learning' is not a factor.

Assumptions 1, 2 and 3 represent one way of handling incompletion situations at stations and is typically considered in the literature. Some of the different approaches to handling incompletions are as follows:

1. A cycle does not start unless all the tasks are completed. In other words, the whole line is stopped until all the tasks are completed.
2. Incomplete tasks are completed at special stations strategically located along the line. Note that the incomplete tasks are completed off the line in our approach.
3. A mobile and qualified team helps the stations having incompletions.

A different approach to handling incompletions is suggested by Silverman and Carter [17]. In their approach, cycle time is extended when certain tasks are not completed within a base cycle time. Hence, the line operates as an unpaced line when such incompletions occur. Tasks are classified into two groups; incompletions of the tasks in the first group are not allowed. A first-group task getting incomplete within the base cycle time is completed even if it requires stopping the whole line for a considerable amount of time. Incompletions of the second-group tasks do not extend the cycle time. As stated by Silverman and Carter [17], it may be economical to stop the whole line for heavy and complex products such as engines, since major disassembly may be required to complete the incompletions off line. Thus, the selection of the approach to handle incompletions should depend on the costs associated with stopping the whole line versus completing the incomplete tasks off line. In addition, some technological constraints and practical reasons may also prohibit the stoppage of the whole line.

Assumption 4 is a direct consequence of the fact that most of the tasks are manually performed and variations in the performance times of these tasks are inevitable. The procedure developed in this paper can be used for processing times that are independently distributed nonnegative random variables with their distribution characterized by a single parameter. The normal distribution is the most frequently assumed distribution for the performance times, see Refs. [2,7–14,17,18], especially if the coefficient of variation is 0.3 or less [20], and, in the experimental work reported below, the processing times are assumed to be nor-

mally distributed random variables truncated at zero with $\sigma_i = a \cdot \mu_i, \quad i = 1, \ldots, N$, where a is a constant. The truncation at zero for a processing time is possible if the probability that the random variable can take on negative values is negligible, see [3]. The assumption that $\sigma_i = a \cdot \mu_i$ would have to be tested empirically in any application. However, it is plausible since the mean and variance of a task processing time is the sum of the means and variances of the processing times of the basic elements that comprise the task. The greater the number of basic elements, the greater will be the mean and variance of the task performance time. Therefore, a large expected processing time of a task implies a greater number of basic elements and consequently a larger standard deviation. The relationship $\sigma_i = a \cdot \mu_i$ has also been assumed by other researchers such as Silverman and Carter [17] and Lau and Shtub [10]. It is not, however, assumed by Kottas and Lau [7–9] who rather assume arbitrary means and variances, thus making their method more general than ours. The comparisons of their method and ours made below are valid only when the relationship $\sigma_i = a \cdot \mu_i$ holds.

Assumptions 5–9 are similar to those typically made for this problem in the literature.

2.1. Cost model used

Incompletion cost, IC_i is the cost of completing task i off the line and is assumed to be greater than $L \cdot \mu_i$ for $i = 1, \ldots, N$. A task is not completed within the cycle time due to two reasons: (i) the task is not completed within C, the operating cycle time, or (ii) the task is a follower of another incomplete task on the precedence diagram. When one task is incomplete, the processing of a set of tasks cannot be started. This set of tasks depends on the precedence diagram and the allocation of tasks to stations.

The objective function is comprised of the total labor and total expected incompletion cost. A general expression is developed in [14] that captures the cost terms of the objective function given the allocation and ordering of the tasks to a given number of stations. The expression of the objective function is as follows:

$$\text{Min } Z = [C \cdot L \cdot K] + \sum_{i=1}^{N} \left[\beta_i \left[\mathrm{IC}_i + \sum_{k \in A_i} \mathrm{IC}_k \right] - \sum_{j=1}^{fs_i} \mathrm{SB}_i^{\,j} \right],$$

where β_i is the probability that the processing of task i is started and not completed within C; A_i is the set of tasks which cannot be started (and hence incomplete) due to the incompletion of task i; fs_i is the total number of starting events of task i and SB_i^j is a correction factor for the overcounted incompletion costs corresponding to starting event j. The processing of task i can be started after all the tasks preceding task i in the station are completed. The tasks preceding task i in the station need not precede task i on the precedence diagram; consequently, several events arise that lead to the starting of task i. The occurrence probabilities of these events are computed with a special probability enumeration tree in which some events are considered several times. These overcounted incompletions costs are subtracted to obtain the exact expected incompletion cost. The expression $[C \cdot L \cdot K]$ represents the total labor cost. The expressions of total expected incompletion cost and labor cost are utilized to determine the design with the minimum cost by varying the number of stations and the allocation of the tasks to these stations.

3. Procedure for determining the number of station and task allocations

The steps of the procedure for determining the number of stations and the allocation of tasks to these stations is as follows:

Step 1. Decompose the problem into subproblems of a given size, say m or fewer tasks.

Step 2. Determine an initial solution for each subproblem by using a dynamic programming (DP) procedure with a bounding strategy [14], hereafter referred to as the truncated DP procedure.

Step 3. Improve the initial solution of each subproblem by using an improvement procedure.

Step 4. Generate the final solution by appending the solutions of the subproblems generated in Step 3.

The first three steps are now considered in detail. The experimentation, conducted with problems for which the solutions of its subproblems are appended to each other, is presented in Section 4.

3.1. Step 1: decomposing the problem

Consider the precedence diagram drawn from 'left' to 'right' with tasks placed as far to the left as possible. The diagram can now be partitioned by columns drawn between each set of tasks occurring from 'left' to 'right'. Activities contained within each column are consequently not linked.

The decomposition procedure arbitrarily combines adjacent columns into groups so that the number of tasks contained therein is a 'reasonable' number —

say, on the order of 20. These groups are serially combined until all tasks in the precedence diagram are considered.

Each group is consequently solved as an independent problem problem 3.2. The decomposition procedure above is expected to result in subproblems with the least number of precedence relations between the tasks. Consequently, the number of feasible sequences that can be generated by the tasks is larger than the number of feasible sequences when the problem is decomposed in any other way. Thus, the final solution obtained with the above procedure is likely to be closer to the optimal one.

3.2. Step 2: determining an initial solution

An initial solution for each group can be generated using the DP procedure with a bounding strategy described in [14]. In the truncated DP procedure, the tasks are allocated station by station following precedence restrictions and the generation of an excessive number of states is avoided by not considering states having a probability of completing tasks below a prespecified value. The truncated DP procedure [14] generates solutions that are almost always as good as those generated by the method of Kottas and Lau [7,8].

3.3. Step 3: improving the initial solution

The truncated DP procedure solution can be improved by using a procedure analogous to a branch-and-bound enumeration tree with the nodes representing station assignments of the tasks. Each level of the tree corresponds to a station. Thus, the tree can have at most N levels corresponding to the N stations with at least one task assigned to each station. The nodes of the next level are formed by considering the precedence constraints. Let CN_i be the cost associated with the assignment of tasks at node i, e be the number of nodes in the tree, and $TCN_i = TCN_{\Gamma_i} + CN_i$ for $i = 1, \ldots, e$, where Γ_i is the parent node of node i. TCN_i represents the cost associated with the assignment of the tasks to node i and all of its ancestor nodes. Note that if node i has no parent node (e.g. the nodes in the first level), then $TCN_{\Gamma_i} = 0.0$ and $TCN_i = CN_i$. Each node has an associated relaxed problem consisting of tasks that are not assigned to either this node or its ancestor nodes; the precedence relations among these tasks are relaxed and the incompletion costs of the tasks are replaced by their cumulative incompletion costs. The relaxed problem is solved with the M-parallel station scheduling procedure outlined in Section 3.3.4. If CRX_i is the cost of the relaxed problem corresponding to node i, then an approximate cost, APP_i, corresponding to this node is defined as follows:

$$APP_i = TCN_i + CRX_i.$$

In the branch-and-bound technique, CRX_i represents a lower bound on the contribution of the remaining tasks for node i. In our case, CRX_i need not be a lower bound, as it is a heuristic solution of the relaxed problem. However, CRX_i is used just like the lower bound value to fathom nodes. Thus, the improvement procedure does not guarantee to obtain the solution with the minimum cost. However, if CRX_i is taken as an ϵ-optimal solution, then it can be shown to generate an ϵ-optimal solution of the original problem in the following sense. Let UB_{cur} denote the current upper bound. Since the solution with the minimum cost belongs to one of the branches of the tree, in the worst case, node i containing the solution with the minimum cost is fathomed subject to

$$APP_i = (1+\epsilon)\,\text{optimal solution value} \geq UB_{cur}$$

$$\text{or, optimal solution value} \geq \frac{UB_{cur}}{1+\epsilon}.$$

Thus, UB_{cur} will be at most within $(1+\epsilon)$ of the optimal solution value.

The advantages of using an ϵ-optimal solution at a node, instead of a lower bound, are that (i) it is easy to obtain and (ii) it is close to the optimal solution value and hence, very effective in cutting down the size of the tree. Of course, the disadvantage is that it can guarantee only ϵ-optimal solutions. The magnitude of the value that one uses for ϵ depends on the performance of the procedure used for solving the relaxed problem at every node of the tree. An experimentation is conducted to investigate the setting of ϵ and is presented in [3].

In the following sections, various features of the improvement procedure are discussed in detail.

3.3.1. The branching scheme

The branching rule for selecting a node to partition the solution space is the best bound rule, that is, the node having the smallest approximate cost is selected to branch.

The scheme used to generate the offspring of a node is identical to that used by Gutjahr and Nemhauser [5] for balancing single model assembly lines. Tasks that are available for assignment are placed in stage 1 and are considered marked. A state is defined as the set of tasks that can be started without prior completion of any other tasks and in any order that satisfies the precedence relations. An immediate follower of a state S of a stage is defined as a task that is an immediate follower of at least one of the tasks in S and is not preceded by any tasks not in S. The unmarked immediate followers of a state are augmented to the current state

to form the states of the next stage. The augmentation of states and corresponding unmarked immediate followers is done in stages. For any state S of stage k, the unmarked immediate followers are placed in a list $F(S)$. Let H be a subset of $F(S)$, then $S \cup H$ is a state for stage $k+1$. For each state of stage k, the unmarked immediate followers are found and placed as marked tasks for stage $k+1$. When all the tasks are marked, the generation procedure is complete. Permutations of tasks are avoided in order to eliminate duplications. The states constitute the offspring nodes of the node being branched. The procedure generates all possible offspring nodes of a parent node, if carried to completion.

Note that (i) if TCN associated with the node $S \cup H_j$ (for some subset H_j of $F(S)$) is larger than UB_{cur}, then all the other nodes formed by $S \cup H_k$, where H_j is a subset of H_k, can be pruned, and (ii) if TCN associated with a node is larger than that of any other descendent node generated previously with the same set of tasks, then this node can be pruned; if TCN is smaller than that of the previously generated node, then the previously generated node can be pruned.

Since each node represents a station, when a node is generated the tasks assigned to that node are resequenced with the single-machine sequencing procedure described next in Section 3.3.2 to further reduce the incompletion cost of the sequence.

3.3.2. Procedure to sequence tasks assigned to a node

The problem of resequencing the tasks assigned to a node can be viewed as the problem of sequencing N tasks on a single-machine with tasks having a common due date (corresponding to the prespecified C) and stochastic processing times. The objective is to sequence tasks in such a way that the expected incompletion cost is minimized. This problem is like a single-machine sequencing problem with a nonlinear loss function, however, the loss function here is defined as the expected incompletion cost.

Consider an arbitrary sequence R in which a pair of adjacent tasks, i and j (j following i), exists such that $IC_i \geq IC_j$. In the sequence R', the positions of the tasks i and j are interchanged. Let $p_i(p_j)$ and $p_i'(p_j')$ denote the incompletion probabilities of task $i(j)$ in sequences R and R', respectively. Also let p_Z be the incompletion probability of the task preceding task $i(j)$ in sequence $R(R')$.

If certain conditions outlined below are met, we can determine the relative order of two adjacent tasks in order to minimize the total expected incompletion cost. Let $\xi_i = p_i - p_Z$, $\xi_i' = p_i' - p_j'$, $\xi_j = p_j - p_i$ and $\xi_j' = p_j' - p_Z$. Also, let $CIC_i(CIC_i')$ be the cumulative incompletion cost of task i in sequence $R(R')$, which consists of the incompletion cost of task i and of those tasks in that station and in subsequent stations that cannot be started due to the incompletion of task i. Thus, $CIC_i = IC_i + \sum_{k \in A_i} IC_k$.

The following theorem states the sequencing rule for two adjacent tasks and the conditions that should be met.

Theorem 1. If $\xi_i CIC_i \leq \xi_i' CIC_i'$ and $\xi_j CIC_j \leq \xi_j' CIC_j'$, then $cost(R) \leq cost(R')$ (that is, task j should be an immediate follower of task i), and if $\xi_i CIC_i \geq \xi_i' CIC_i'$ and $\xi_j CIC_j \geq \xi_j' CIC_j'$, then $cost(R) \geq cost(R')$ (that is, task i should be an immediate follower of task j).

Proof. $cost(R) = \xi_i CIC_i + \xi_j CIC_j$ and $cost(R') = \xi_j' CIC_j' + \xi_i' CIC_i'$. Therefore, $cost(R) - cost(R') = (\xi_i CIC_i + \xi_j CIC_j) - (\xi_j' CIC_j' + \xi_i' CIC_i')$. Since $\xi_i CIC_i \leq \xi_i' CIC_i'$ and $\xi_j CIC_j \leq \xi_j' CIC_j'$, it follows that $cost(R) \leq cost(R')$. The second part of the theorem can be proved similarly.

Note that only tasks i and j are considered in $cost(R)$ and $cost(R')$. The tasks preceding task $i(j)$ and the ones following task $j(i)$ in sequences $R(R')$ also contribute to $cost(R)$ and $cost(R')$. However, these contributions can be ignored, since they are equal to each other in sequences R and R'. Note also that the precedence relations among tasks are ignored in the above theorem. If $j \in A_i$, it is obvious that sequence R' in which task i following task j violates the precedence relations. Thus, such interchanges are not permitted.

The sequencing procedure first orders the tasks in the descending order of their cumulative incompletion costs with the precedence relations being observed. If the order of the tasks violates the precedence relations, then the positions of those tasks are changed accordingly, though the sequence of the tasks no longer remains in a descending order of the cumulative incompletion costs. Then, the conditions of theorem 1 are applied to each pair of adjacent tasks in the sequence to arrange them accordingly. If $j \in A_i$, sequence R' in which task i following task j violates the precedence relation and the interchange is not permitted. In certain regions of the sequence, the incompletion probabilities of tasks i and j cannot be differentiated; they are both assumed to be either negligible or equal to unity due to the asymptotic shape of the normal distribution.

If tasks i and j are in a region in which the incompletion probabilities can be differentiated and the conditions of theorem 1 are not met, then the relative order of the tasks is determined by comparing $cost(R)$ with $cost(R')$. If $cost(R) \leq cost(R')$, then task j follows task i; otherwise, task i follows task j. In summary, the following three situations arise while determining the

Table 1
Parameters of the example problems

Example	Number of tasks	*F*-ratio	Cycle time (min)	Number of stations			RAN_2
				Truncated DP procedure	Technique of Kottas and Lau [9]	Improvement procedure	
1	11	0.000	58.4	9	9	8	0.083
2	11	0.000	72.2	7	7	7	0.076
3	11	0.000	99.9	7	7	3	0.062
4	11	0.491	23.8	7	8	2	0.056
5	11	0.491	37.6	5	6	5	0.099
6	11	0.491	51.4	6	6	6	0.092
7	11	0.418	65.3	8	9	6	0.085
8	11	0.418	79.1	7	9	7	0.079
9	11	0.418	93.0	9	9	7	0.072
10	11	0.800	57.9	7	8	4	0.068
11	11	0.800	92.4	7	7	2	0.058
12	11	0.800	50.8	7	7	5	0.090
13	15	0.000	99.2	9	9	6	0.073
14	15	0.000	57.6	10	10	5	0.055
15	15	0.000	16.0	10	10	10	0.088
16	15	0.257	64.4	8	8	6	0.077
17	15	0.257	22.8	10	10	2	0.053
18	15	0.257	71.2	9	10	9	0.085
19	15	0.781	29.6	10	10	10	0.068
20	15	0.781	78.0	8	9	2	0.050
21	15	0.781	15.6	11	11	11	0.099
22	16	0.575	84.8	6	7	6	0.065
23	16	0.575	43.2	8	10	8	0.098
24	16	0.575	91.6	9	9	9	0.080
25	17	0.382	50.0	9	10	6	0.063
26	17	0.382	98.4	10	11	10	0.095
27	17	0.382	56.8	9	10	9	0.078
28	18	0.379	15.2	11	11	2	0.060
29	18	0.379	63.6	7	8	7	0.093
30	18	0.379	22.0	11	12	11	0.075

relative order of two adjacent tasks assigned to the same node of the tree formed by the improvement procedure.

Situation 1. The adjacent tasks meet the conditions of theorem 1. The relative order of the tasks is determined according to theorem 1.
Situation 2. The adjacent tasks do not meet the conditions of theorem 1 and their incompletion probabilities can be differentiated. The relative order of the tasks is determined by comparing the expected incompletion costs of the sequences corresponding to the two positions of the tasks.
Situation 3. The adjacent tasks do not meet the conditions of theorem 1 and their incompletion probabilities cannot be differentiated. The optimal relative order of the tasks cannot be determined. Consequently, no action is taken and the task with the larger cumulative incompletion cost remains as the first task.

3.3.3. Node evaluation scheme

When a node is generated, the tasks in the node are resequenced as discussed in the previous section, and the cost CN associated with the sequence is computed. CN consists of the labor cost of a station and the expected incompletion costs of the tasks in the node. The expression for CN_i corresponding to node i is similar to the objective function expression in Section 2.1.

In order to compute CRX_i, let G_i be the set of tasks in node i and in all its ancestor nodes. The set of tasks in the relaxed problem corresponding to node i is given by the set $W–G_i$. Let N_{ir} be the number of tasks

Table 2
Comparative analysis between the technique of Kottas and Lau [9], the truncated DP procedure and the proposed improvement procedure

Example	Solution values obtained using			Percentage difference between proposed improvement procedure and	
	Technique of Kottas and Lau[a]	DP procedure[a]	Improvement procedure[a]	Kottas and Lau [9]	DP procedure
1	34.420	34.420	32.539	5.5	5.5
2	25.372	25.372	25.372	0.0	0.0
3	35.842	35.842	30.496	14.9	14.9
4	9.880	8.833	7.963	19.4	9.9
5	11.810	9.930	9.930	15.9	0.0
6	15.576	15.487	15.487	0.6	0.0
7	41.202	37.938	36.542	11.3	3.7
8	50.301	44.041	44.041	12.5	0.0
9	42.873	42.873	41.556	3.1	3.1
10	20.993	20.993	20.240	3.6	3.6
11	33.264	32.595	26.049	21.7	20.1
12	19.946	19.937	19.741	1.0	1.0
13	62.104	61.753	55.109	11.3	10.8
14	28.828	28.806	25.433	11.8	10.7
15	10.064	10.063	10.063	0.0	0.0
16	25.799	25.766	25.713	0.3	0.2
17	11.434	11.430	9.388	17.9	17.9
18	36.375	32.107	32.107	11.7	0.0
19	15.462	15.880	14.938	3.4	5.9
20	37.575	33.811	30.076	20.0	11.1
21	8.831	8.644	8.644	2.1	0.0
22	33.007	29.038	29.038	12.0	0.0
23	22.537	19.348	19.348	14.2	0.0
24	52.219	52.219	52.219	0.0	0.0
25	31.654	29.114	27.588	12.9	5.2
26	75.102	71.919	71.919	4.1	0.0
27	28.734	25.928	25.928	9.8	0.0
28	8.580	8.563	7.159	16.6	16.4
29	30.085	27.246	27.246	9.4	0.0
30	14.851	13.688	13.688	7.8	0.0

[a] Solution values are in US$/unit.

in the set $W–G_i$. In a relaxed problem, precedence constraints are relaxed, but the incompletion costs of the tasks are replaced by their cumulative incompletion costs. This relaxed problem is solved as an M-parallel station scheduling problem with the procedure outlined in Section 3.3.4. The cost of the relaxed problem solution associated with node i is taken as a lower bound on the contribution of the tasks in $W–G_i$. Let CIN_{iM} denote the expected incompletion cost of the relaxed problem solution for node i obtained by the M-parallel station scheduling procedure. Then, the total expected cost of the relaxed problem, CRX_i, may be expressed as follows:

$$\mathrm{CRX}_i = \min_{M=1,\ \ldots,\ N_{ir}} C \cdot L \cdot M + \mathrm{CIN}_{iM} \text{ for } i = 1,\ \ldots,\ e_l.$$

Note that in order to find a good estimate of CRX_i, the M-parallel station scheduling procedure must be applied N_{ir} times to the relaxed problem corresponding to node i for $M = 1, \ldots, N_{ir}$. This, in reality, can be very time consuming. In order to limit the range of this search over the number of stations, an experiment was conducted to compare the number of stations obtained by the above procedure with the search at each node applied N_{ir} times, to that obtained by the truncated DP procedure with the bounding strategy [14] and the method of Kottas and Lau [9]. The data for the test problems are generated as follows: $\mu_i \sim U[0;\ C]$, $\sigma_i = \mathrm{RAN}_1 \mu_i$ and $\mathrm{IC}_i = \mathrm{RAN}_2 \mu_i$ for $i = 1, \ldots, N$ and $L = 0.05$ US$/min. $C \sim U[10;\ 100]$, $\mathrm{RAN}_1 \sim U[0.04;\ 0.06]$ and $\mathrm{RAN}_2 \sim U[L;\ 2L]$. Table 1 depicts the number of tasks, flexibility-ratios (F-ratio),

cycle times, number of stations in the solutions of the problems using the truncated DP procedure [14], the method of Kottas and Lau [9] and the improvement procedure. *F*-ratio is a measure of the number of feasible sequences that can be generated and takes values from 0.0 to 1.0. The value of 1.0 corresponds to a problem with no precedence relations whereas the value of 0.0 corresponds to a problem with tasks ordered serially. The difference in the number of stations in the three solutions depends on relationships between the cycle time, task processing times and the relative magnitude of the incompletion costs. As seen in Table 1, for 16 of the 30 problems, the difference between the number of stations determined by the truncated DP procedure ($NSTA_{DP}$) and the improvement procedure is less than or equal to 1; the difference, on the average, is 2.00. In all cases, the number of stations obtained by the improvement procedure is less than or equal to $NSTA_{DP}$. Half of the Kottas and Lau [9] solutions require more stations than $NSTA_{DP}$, and are consequently generated at a higher cost. Table 2 gives the actual incompletion costs using the three methods, together with percentage differences in comparison with the improvement procedure. It is seen that the proposed method dominates the method of Kottas and Lau [9] in every comparison, while it improves on the truncated DP procedure solution in about half the cases and equals the DP solutions for the remainder. (Recall, however, both have and below, that the method of Kottas and Lau allows distributions in which means and standard deviations are not related, and the comparisons are made under our somewhat more restrictive assumption about distributions of task performance times.) We use these results to guide the search over the number of stations to be used at every node to compute CRX_i. If node i is at level j, then the corresponding relaxed problem is expected to have $NSTA_{DP}-j$ stations. Thus, the above procedure is applied to the relaxed problem for the number of stations in the neighborhood of $NSTA_{DP}-j$. Based on the assumption that the difference between the numbers of stations obtained by the truncated DP and the improvement procedures is less than 5 (a conservative number based on the above experimentation), the procedure is applied to the relaxed problem associated with node i of level j 11 times for $M = NSTA_{DP}-j-5, \ldots, NSTA_{DP}-j+5$.

Next, we describe an *M*-parallel station scheduling procedure for the relaxed problem. The term parallel here implies that the stations are assumed to be in parallel rather than in series. Since this procedure is used to solve the relaxed problem at every node of the enumeration tree of the improvement procedure, the performance of this procedure affects the performance of the improvement procedure significantly. The closer the solution of this procedure is to the optimum, the better will be the quality of the final solution of the problem.

3.3.4. M-parallel station scheduling procedure for the relaxed problem

This procedure is based on the single-processor sequencing procedure presented in Section 3.3.2 and the steps are as follows:

> Step 1. (Construction of the single-station sequence). Set the due date equal to $M \cdot C$. Rank the tasks in the descending order of their cumulative incompletion costs. Examine each pair of adjacent tasks to find the situation described in Section 3.3.2, and determine the relative order of the tasks. Continue until all pairs of adjacent tasks are examined.
>
> Step 2. (Allocation of the tasks to M stations). Allocate the tasks to M machines sequentially in their order of appearance in the single-station sequence by assigning the next task to the machine that has the least sum of the expected processing times of the tasks already assigned to it. Continue until all the tasks are assigned.
>
> Step 3. (Resequencing of the tasks on the stations). The tasks assigned to each machine are resequenced according to the single-station sequencing procedure.

The *M*-parallel station scheduling procedure presented above does not guarantee the optimal solution. Let the solution obtained using this procedure at every node be an ϵ-optimal solution, i.e. it is within $(1+\epsilon)$ of the optimal solution. Also, if CRX_i is the total expected cost obtained using this procedure by its application to the relaxed problem associated with node i, then there exists a ρ_i, $\rho_i \geq 0$, such that $CRX_i/(1+\rho_i)$ is a valid lower bound for node i. Consequently, if $CRX_i/(1+\rho_i)$ is used instead of APP_i at node i, then the enumeration procedure will guarantee the optimal solution. Now, ϵ and ρ_i are not known apriori. However, if ρ_i is taken to represent the ρ_i-optimality of the scheduling procedure used to solve the relaxed problem at node i, then $\epsilon = \max_{i=1,\ldots,e} \rho_i$, since, in the worst case, one of the nodes leading to the optimal solution can have the maximum ρ value. This suggests the following way to estimate the value of ρ experimentally. A problem is solved using the same ρ value for all nodes in the enumeration tree of the improvement procedure. If the ρ value assumed is not sufficiently large, then increasing its value should improve the solution. Since the procedure is heuristic and approximate costs are used instead of upper bounds for pruning nodes, increasing the value of ρ does not necessarily improve the solution, but, on the average, an improvement in the solutions is expected. If the improvement in the solution stabilizes beyond a ρ

Table 3
Ninety percent confidence interval limits on percentage differences between the solution values of improvement procedure, technique of Kottas and Lau [9] and truncated DP procedure

RAN2	Ninety percent confidence interval limits on percentage solution values of difference between the improvement procedure and	
	Kottas and Lau solution	Truncated DP procedure solution
L	[20.7; 24.4]	[16.3; 20.8]
$U[L, 2L]$	[7.5; 11.7]	[2.7; 6.7]
$2L$	[4.0; 7.2]	[0.2; 1.3]
$3L$	[3.0; 5.9]	[−0.1; 1.2][a]
$4L$	[2.5; 5.3]	[−0.3; 1.0][a]

[a] Note that the negative value just indicates the lower limit of the 90% confidence interval; the actual values obtained are all non-negative.

value, it implies that the value of ρ is either larger than ϵ of the scheduling procedure or very close to it. An experimentation was conducted to estimate the values of ρ and ϵ and is reported in Ref. [3]. The increase in improvement was found to stabilize, on average, at $\rho \geq 0.1$, with the M-parallel station scheduling procedure generating solutions within 110 percent of the optimal solution. Consequently, the M-parallel station scheduling procedure generates almost optimal solutions for this ρ value.

Note that for some examples in Table 1 the difference between the number of stations generated by the improvement procedure and those generated by the method of Kottas and Lau [9] and the truncated DP procedure is significant. To explain this, observe the last column of Table 3 which lists the value of RAN_2 used for different examples. RAN_2 is a random variable generated from a uniform distribution between L and $2L$. As the labor rate, L, is taken as 0.05 US$/min; RAN_2 is distributed uniformly between 0.05 and 0.10. For all the examples with large differences in numbers of stations (namely, 3, 4, 11, 14, 17, 20, 28), the RAN_2 value is relatively small. That means, for small incompletion cost, the method of Kottas and

Table 4
Results of experimentation with problems of greater than 20 tasks

Example	Number of tasks	F-ratio	Cycle time (min)	Kottas and Lau solution (min)	Approximation procedure		
					Solution value (US$/unit)	CPU time taken (s)	Percentage improvement over Kottas and Lau solution
31	30	0.372	35.6	35.655	33.126	5.73	7.1
32	30	0.372	84.0	82.222	78.872	9.96	4.1
33	30	0.372	42.4	61.452	61.338	124.83	0.2
34	40	0.173	90.8	166.947	140.716	147.02	15.7
35	40	0.173	49.2	75.875	70.952	48.33	6.5
36	40	0.173	97.6	150.763	150.639	254.87	0.1
37	30	0.051	56.0	78.858	70.371	62.55	10.8
38	30	0.051	14.4	16.883	16.251	28.55	3.7
39	30	0.051	62.8	46.728	45.155	11.12	3.4
40	30	0.147	76.4	84.470	78.420	19.47	7.2
41	30	0.147	34.8	46.062	42.578	124.79	7.6
42	30	0.147	83.2	90.035	78.162	131.91	13.2
43	50	0.074	41.7	97.969	95.127	50.60	2.9
44	50	0.074	90.0	226.199	223.261	13.26	1.2
45	50	0.074	48.5	106.564	90.746	156.22	14.8
46	60	0.057	96.9	301.845	292.288	131.23	3.2
47	60	0.057	55.3	142.366	141.807	251.05	0.4
48	60	0.057	13.7	29.130	26.427	380.79	9.3

Lau [9] and the truncated DP procedure utilize more stations than needed to minimize the labor and the expected incompletion costs. However, as the relative magnitude of IC increases, the number of stations utilized in the solutions generated by these procedures tend to be close to those utilized by the improvement procedure. This is also shown in Table 3 which depicts percentage differences between the solution values of the improvement procedure and those of the method of Kottas and Lau [9] and the truncated DP procedure. We have varied the value of RAN_2 from $4L$ to L, namely from 0.2 to 0.05, and solved the problems with the three procedures. The solution value of the procedure developed in this paper is significantly superior to the ones obtained by the method of Kottas and Lau [9] and the truncated DP procedure at the lower values of RAN_2. This indicates that the method of Kottas and Lau [9] and the truncated DP procedure tend to be insensitive to the relative magnitude of the incompletion costs, whereas the procedure developed in this paper gives solutions that are very close to the optimal ones for all relative magnitudes of incompletion costs.

4. Experimentation with problems requiring decomposition

A comparison of the results obtained by the procedure developed in this paper to those obtained by the method of Kottas and Lau [9] and the truncated DP procedure [14] was presented in Tables 1 and 2. These problems contained at most 20 tasks. Next, we consider problems with 30–60 tasks. These problems are decomposed into subproblems of 20 or fewer tasks as described in Section 3.1. Table 4 summarizes information about the problems and the results obtained. Columns 2, 3 and 4 depict the number of tasks, F-ratios and the cycle time of the problems considered. The solution values obtained by the method of Kottas and Lau [9] and the proposed procedure are shown respectively in columns 5 and 6. The CPU time limit on the improvement procedure of each subproblem was set to 120 s on an IBM 3031. The total CPU time taken by the procedure is depicted in column 7. The computational requirements for the Kottas and Lau [9] procedure are considerably smaller than those reported in column 4, however, the decreasing cost of computer usage lessens the importance of this issue. Moreover, within reasonable computation time, it seems more important to determine better quality solutions as the production cost saving will be orders of magnitude larger than the cost of computations. The last column depicts the percentage improvement of the proposed procedure over the solution of the method of Kottas and Lau [9]. On average, the approximation procedure generated designs whose total system costs are 6.2% lower than those of the designs generated by the method of Kottas and Lau [9].

5. Concluding remarks

A comprehensive procedure for solving the stochastic, single-model assembly line balancing problem was developed and presented in this paper. This approach begins with the truncated DP procedure solution, and uses a branch-and-bound type of procedure to generate an improved solution. Experimental results indicate that better solutions are generated than those obtained by the method of Kottas and Lau [9] within reasonable computation times for the single model, stochastic line balancing problem under the condition described in Section 2, including specifically that the means and standard deviations of task performance times are linear. When improvements occur, the solutions are always associated with fewer stations and are shown to generate designs with smaller cost values. Indeed, solutions using the proposed method, tested over a wide range of parameter values, were shown to either equal to or dominate those obtained by the truncated DP method, which in turn, equals or dominates those generated by the Kottas and Lau procedure [9].

References

[1] Arcus AL. COMSOAL: a computer method of sequencing operations for assembly lines. International Journal of Production Research 1966;4(4):259–77.
[2] Chakravarty AK, Shtub A. A cost minimization procedure for mixed model production lines with normally distributed task times. European Journal of Operational Research 1986;23(1):25–36.
[3] Erel E. A methodology to solve single-model, stochastic assembly line balancing problem and its extensions. Unpublished doctoral dissertation, 1987.
[4] Freeman DR, Jucker JV. The line balancing problem. The Journal of Industrial Engineering 1967;18(6):361–4.
[5] Gutjhar AL, Nemhauser GL. An algorithm for the line balancing problem. Management Science 1964;11(2):308–15.
[6] Ignall E. A review of assembly line balancing. Journal of Industrial Engineering 1965;16(4):244–54.
[7] Kottas JF, Lau HS. A cost oriented approach to stochastic line balancing. AIIE Transactions 1973;5(2):164–71.
[8] Kottas JF, Lau HS. A total operating cost model for paced lines with stochastic task times. AIIE Transactions 1976;8(2):234–40.
[9] Kottas JF, Lau HS. A stochastic line balancing procedure. International Journal of Production Research 1981;9(2):177–93.
[10] Lau HS, Shtub A. An exploratory study on stopping a

paced line when incompletion occur. IIE Transactions 1987;19(4):463–7.

[11] Moodie CL, Young HH. A heuristic method of assembly line balancing for assumptions for constant and variable work element times. The Journal of Industrial Engineering 1965;16(1):23–9.

[12] Ramsing K, Downing R. Assembly line balancing with variable element times. Industrial Engineering, January, 1970, pp. 41–43.

[13] Reeve NR, Thomas WH. Balancing stochastic assembly lines. AIIE Transactions 1973;5(3):223–9.

[14] Sarin SC, Erel E. Development of the cost model for the single-model, stochastic assembly line balancing problem. International Journal of Production Research 1990;28(7):1305–16.

[15] Sculli D. Short term adjustments to production lines. Computers and Industrial Engineering 1984;8(1):53–63.

[16] Shtub A. The effect of incompletion cost on the line balancing with multiple manning of work stations. International Journal of Production Research 1984;22(2):235–45.

[17] Silverman FN, Carter JC. A cost-based methodology for stochastic line balancing with intermittent line stoppages. Management Science 1986;32(4):455–63.

[18] Silverman FN, Carter JC. A cost-effective approach to stochastic line balancing with off-line repairs. Journal of Operations Management 1984;4(2):145–57.

[19] Vrat P, Virani A. 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 1976;14(4):445–63.

[20] Wilhelm WE. On the normality of operation times in small-lot assembly systems: a technical note. International Journal of Production Research 1987;25(1):145–9.