ISSN 2305-8277 (Online), 2012, Vol. 1, No. 3, 78-95.

ON ALLOCATION OF TASKS WITH DIFFERENT LEARNING SLOPES TO STATIONS IN ASSEMBLY LINE FOR LOTS[†]

YUVAL COHEN*
The Open University of Israel

ABSTRACT

This paper addresses the problem of allocating work elements with various learning slopes to stations to minimize the makespan of many products produced in an assembly line for lots. The lots are characterized by a low overall demand for each product. There is no buffer permitted in between the stations, and the line operates under learning. Due to the nature of work, station's learning slopes can be different. We propose a two stage optimization methodology: the first stage is an optimization based on a non-linear formulation for work allocation with some constraints relaxation; the second stage drops the relaxations and finds a solution that is the closest to the unconstrained solution found in the first stage. We show that in the presence of learning, the optimal makespan requires assigning different workloads to different stations. This difference depends on the number of cycles in the lot, the station's learning slope, and the station's location along the line. The savings in the optimal makespan value due to the imbalanced loading of work over the balanced loading case are demonstrated.

Keywords: Work Allocation; Learning; Batch Assembly; Assembly line.

INTRODUCTION

This paper considers makespan minimization of a batch (or lot) of products processed sequentially on a mostly manual assembly line whose work stations have different learning slopes. The situation of processing lots of products with low overall demand in line configuration can typically be attributed to science-based products for which the total demands are likely to be between the mid tens to the low hundreds. Practical instances of this setting are presented in Globerson and Shtub (1984), Chakravarty (1988), Dar-El et al. (1995b), Cohen and Dar-El (1998), Dar-El (2000), Ardity et al. (2001), and Cohen et al. (2006, 2008).

Although there is bounty of literature on assembly line balancing (ALB) on the one hand and on the learning phenomenon on the other hand, the literature on the intersection of both subjects is scarce. Especially the literature on optimizing work allocation in assembly lines under learning is very limited. Since we will show that balancing policy is not optimal the introduction will concentrate on presenting the setting and assumptions of the model, the

[†] The views or opinions expressed in this manuscript are those of the author(s) and do not necessarily reflect the position, views or opinions of the editor(s), the editorial board or the publisher.
* Corresponding author Email: yuvalco@openu.ac.il

importance of industrial learning to the model and presenting relevant background in industrial learning calculations.

Setting and Assumptions of the Model

The assembly or production process is composed of many tasks, each with known standard time, learning slope and precedence constraints. Each station is assigned a combination of these tasks, and there is no buffer permitted in-between the stations. Cohen and Dar-El (1998) presented a specific instance of this problem where all the stations have the same learning slope. Cohen et. al. (2006) showed that even under homogeneous learning, allocating the same amount of work to each station is not optimal. In the current paper, the stations may have different learning slopes based on their work content. We also assume that the number of stations, L, is known a priori (a method to compute optimal L is given in Cohen and Dar-El (1998)).

Since maximizing the simultaneous work in all stations minimizes the makespan, and since simultaneous work is best achieved when the learning slopes of all stations are the same, it would be ideal to have the same learning slope at all stations (see Cohen et. al. (1998)). In reality however, learning slopes may vary considerably between stations. This may be due to highly constrained networks, or segregated tasks with very dissimilar learning slopes, or due to existence of a bottleneck in the network of tasks. In this paper we show that the optimal workload assignment to stations is considerably effected by the learning phenomenon that induces variability between stations.

Industrial Learning Calculations

According to the mathematical form of the classical power learning curve (since Wright (1936)), the cycle time reduces by a constant percentage (ϕ) every time the quantity produced is doubled (ϕ is called the learning slope). We define cycle time to be the time taken to process the work allocated to a station. If t_1 is the execution time of the first cycle, and t_n is the execution time of the n^{th} cycle, then the learning curve can be expressed as follows:

$$t_n = t_1 \cdot \phi^{\log 2(n)} \qquad (1)$$

A more prevalent form of Equation (1) is based on the classic learning constant (b) (Yelle (1979)):

$$b = -\log_2(\phi) \qquad (2)$$

to give:

$$t_n = t_1 \cdot n^{-b} \qquad (3)$$

A prevalent approximation of the time for the first ***n*** cycles of a learning curve is derived by taking the integral of equation 3 as shown in equation 4

$$T_n = \left(\frac{t_1}{1-b}\right) \cdot n^{(1-b)} \qquad (4)$$

The effect of learning on production scheduling has been analyzed by several researchers (see, for example: Adler and Nanda (1974), Kroll (1989), and Dar-El and Rabinovitch (1988), and Cohen and Dar-El (1998)), who considered the simple case where learning is identical for all stations in the line. Chakravarty (1988) considered the case of no-buffers and attributed different learning curve parameters to each station (by using the tasks in the precedence diagram). The minimization of makespan in assembly lines under the constraint of low total demand and learning was tackled by Globerson and Shtub (1984).

They proposed *balancing* the line with the estimated task times of the mid-lot cycle. However, in this paper, we show that optimal work allocations to stations should not be identical even when learning slopes are identical in all stations, so *balancing* is not an optimal policy.

Further work in the Learning Curve research has increased our understanding of the learning phenomenon and prediction of the two learning parameters (t_1, b). A Thorough review of the subject could be found in Dar-El (2000). In particular, if *STD* is the standard time of a task, then it is shown that t_1, the time for the first cycle, is approximately expressed by:

$$t_1 = STD \cdot (57 - 60\phi) \qquad (5)$$

For example, with an 80% learning slope (ϕ=80%), the first cycle time (t_1) is 9 times its standard time. Indeed, the learning slope and the standard time of any task could be estimated: the standard time can be determined by using the Predetermined Motion Time Standards (PMTS), and the learning slope ϕ could be estimated by a method described in Dar-el (2000) that knowing the learning slope, enables the use of the ratio of the first cycle to its standard time defined as G(ϕ):

$$G(\phi) = \frac{t_1}{STD} = 57 - 60 \cdot \phi \qquad (6)$$

For example, G(ϕ=80%) = 57-60*0.8 = 9. Equation 6 enables using the equation $t_1 = STD \cdot G(\phi)$ for computing the first cycle time of any task or combination of tasks.

The work allocated at each station is comprised of a set of tasks, some of which simple and well are defined, while others are quite complex, requiring constant reference to specifications or manuals for their execution and performance. It is now recognized that simpler tasks are associated with slower learning rates (or higher learning slopes ϕ) meaning, there is less to learn in order for an operator to become adept at executing this type of work (Dar-El (2000)). Thus, we have the situation where learning slopes may vary from 70% for high complexity tasks, to 90% for low complexity tasks (Gilad et al. (1991), Dar-El (2000)).
The rest of the paper is structured as follows: Section 2 presents theoretical considerations of the problem which serves as a basis for understanding this paper. Section 3 describes the optimization framework and the data processing requirements for the optimization input. Section 4 describes the non-linear programming formulation, analyzes its savings when compared with equal work allocations to all stations, and discusses its characteristics and the factors affecting the results. Section 5 considers a more realistically constrained problem and describes how MUST (Multiple Solution Line Balancing Technique) could be modified to find a near optimal solution. Section 6 concludes the paper.

THEORETICAL CONSIDERATIONS

Starvation and Blockage under Varying Learning Slopes

Starvation and blockage are two forms of idle time in a workstation (see Dar-El and Mazar (1989)). Starvation occurs when a workstation must wait for the next product to arrive from its preceding station. Blockage occurs when a station finishes processing its work, but cannot pass the product to the next station. Learning may cause both blockage and starvation to occur among the various stations. The order of the stations along the line determines when the blockage and starvation occur as could be seen from two cases depicted in figures (1.a) and (1.b).

FIGURE 1.a

Case A: Blockage and Starvation in a two station line with steeper learning slope at the second station

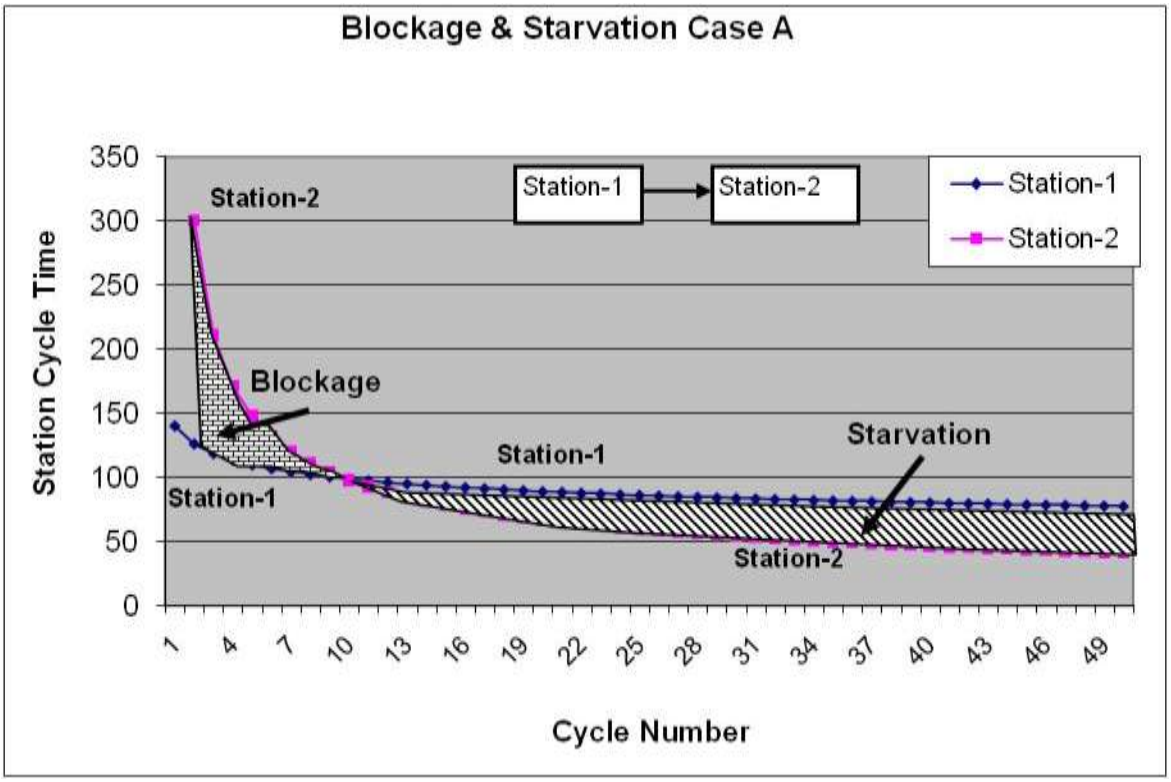
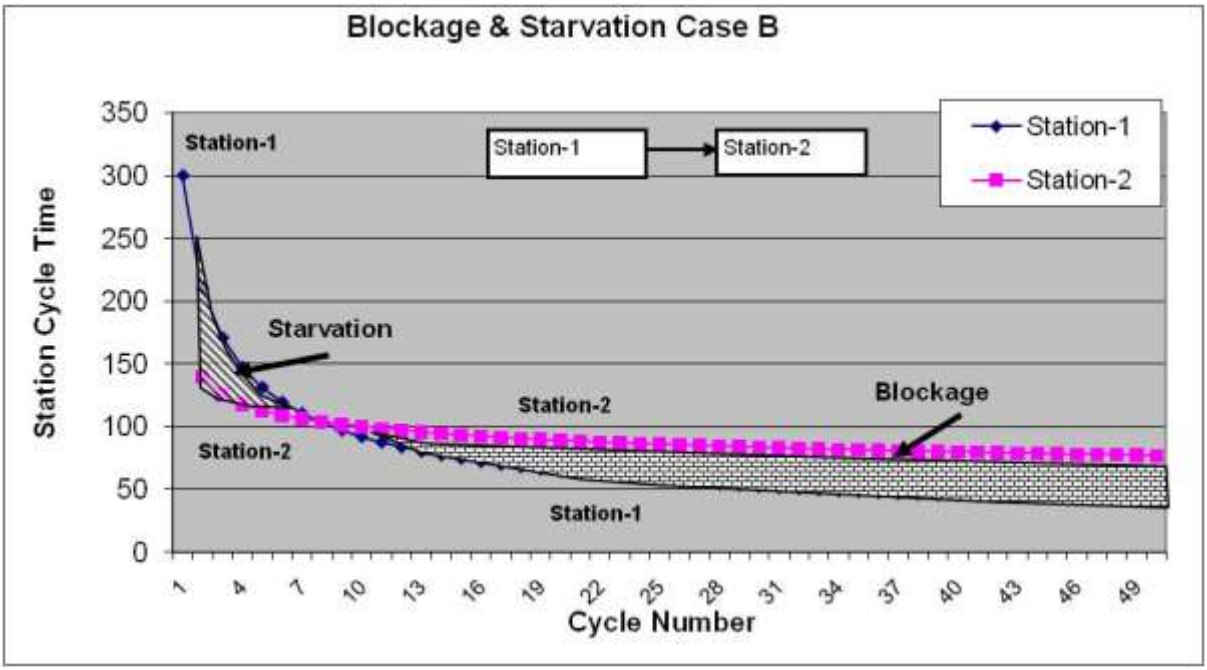


FIGURE 1.b

Case B: Starvation and Blockage in a two station line with steeper learning slope at the first station



The upper envelope, its structure, and the concept of total-dominance

As a consequence of the starvation and blockage, one can easily verify that the actual cycle time of both stations in figures 1.a and 1.b is given by the upper envelope of the two learning curves (the difference between the upper envelope and a lower curve is either starvation or blockage).

When there are no buffers, the upper envelope of the stations learning curves gives the actual production curve for any number of stations. The area below the upper envelope is the total makespan.

Definition:

Segment. We define a segment of a station's learning curve in the upper envelope to be the continuous part of the upper envelope that belongs to the learning curve of that station (it is also the period in which the station is a bottle neck).

Theorem 1. Drawing any upper envelope of the stations' learning curves, the order of *segments* (left to right) is from the leftmost steepest station's segment of the upper envelope (highest b and smallest ϕ) to the rightmost flattest station's segment (smallest b and highest ϕ).

Thus, for any pair of stations i,j - if a segment of station i appears in the upper envelope before the segment of station j, theorem 1 states that $b_i \geq b_j$.

Proof. Define a pair of stations i, j with learning curve segments in the upper envelope, such that the segment of station i appears before the segment of station j in the upper envelope. Moreover, define their corresponding learning constants to be b_i and b_j. When the segment of station i appears as part of the upper envelope, its cycle times are longer than any other station's cycle times (including station j's cycle times).

Any case contradicting Theorem 1 must have at least one pair of stations i, j where $b_i < b_j$.

However, if bi<bj then the cycle times of station j are decreasing in a greater rate than the cycle times of station i. Since station j started with smaller cycle times than those of station i, and its cycle times decrease in a faster rate, it will never have cycle time greater then the corresponding cycle time at station i. Under the above definitions, if $b_i < b_j$, a learning curve segment of station j will never appear in the upper envelope, causing a contradiction to the very definition of station j.

Therefore, if a segment of station i appears before the segment of station j, in the upper envelope, then $b_i \geq b_j$ (QED).

Definitions:

When the cycle time of station i is greater then the time for the same cycle of station j we say that station i dominates station j in that cycle.

Dominance in an assembly line occurs when during a specific time interval, a dominant station has longer cycle times then all other stations. Clearly, the dominant station is a bottle-neck. But because such a dominant station may improve (learn) faster then other stations, its cycles may become shorter until they are shorter then some other station's cycles. At this stage the other station with the longest cycle assumes the role of bottleneck. It is clear that the bottleneck can be dynamic: at different periods there are different stations having the role of a bottle-neck station. It follows that the cycles at the bottleneck dictates the pace of production

for the whole line and the makespan can be calculated by summing the Dominance periods of all bottleneck stations.

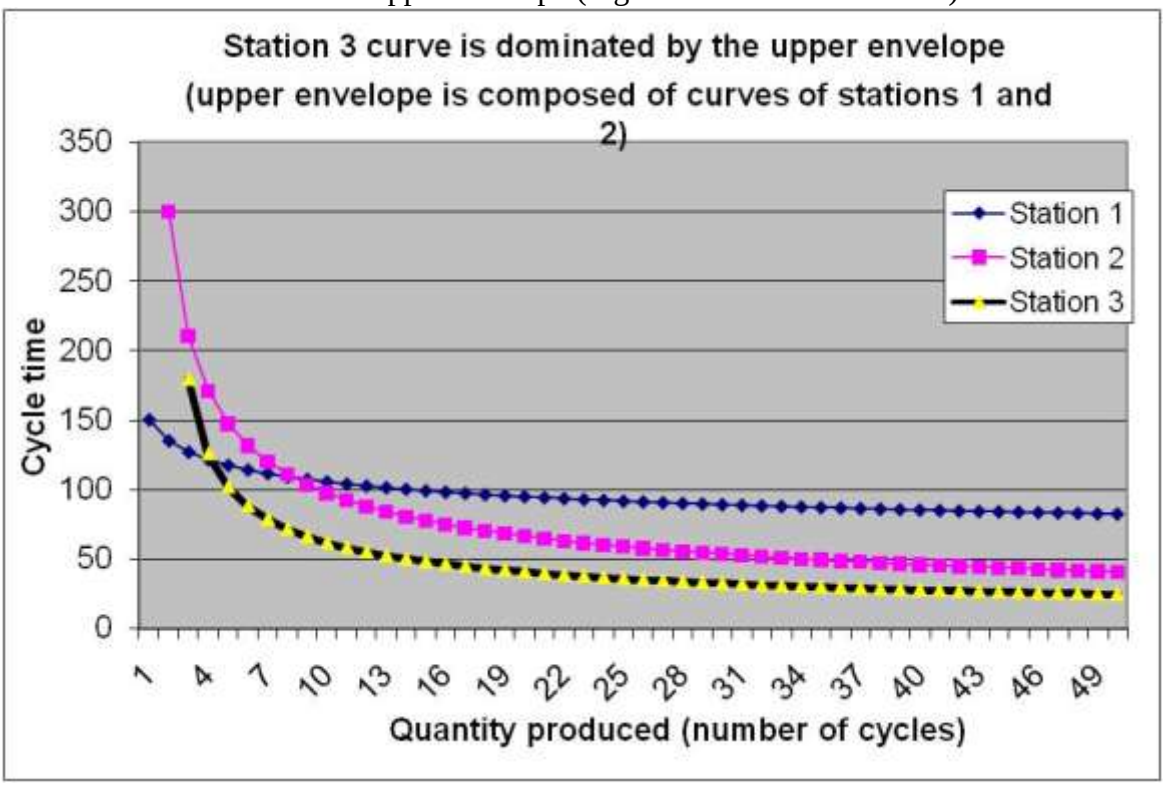
Total-dominance occurs when a station has all of its cycle times below the upper envelope we say that the station is Totally Dominated. Any transfer of work from a station that participates in the upper envelope to a totally dominated station, results in a makespan reduction. Thus, a minimal makespan cannot allow Total-Dominance. This implies that at the optimum, every station, at some stage, must have a segment in the upper envelope. This, in turn, implies that each station will have two intersection points on the upper envelope - marking the start and finish cycles of the period when the particular station was the bottleneck (for the first bottleneck station the first intersection point defined as zero (0) and for the last bottleneck the second intersection point defined as M = the total demand).

Characterizing the Optimal Upper Envelope

At the optimum, every station must, have a segment in the upper envelope. Otherwise, stations that do not appear in the upper envelope could be assigned some work elements of stations that do appear in the upper envelope, thereby reducing the makespan and violating the axiom that optimum cannot be improved.

FIGURE 2

Total-Dominance of the upper envelope (segments of stations 1 and 2) over station 3



Combining theorem 1 and the above logic (for optimality of a segment for each station) we get:

Theorem 2. The only way to get an upper envelope comprised of segments of all stations, is to sequence the bottleneck periods of the stations according to a decreasing order of learning constants b (or increasing order of ϕ). It means that the first bottleneck would have a steeper learning curve than the second and so on.

Since the objective is to minimize the makespan, and since the makespan is the area below the upper envelope, we need to minimize this area. Moreover, we know that in the optimum all stations have segments in the upper envelope ordered according to decreasing learning constants b_i .

Characterizing a station based on its allocated tasks

The act of producing a product is broken down to small work elements we call tasks, and these tasks are assigned to the various stations. The standard time of the tasks can be derived using a Predetermined Motion Time Standard (PMTS) technique (e.g. MTM or MOST). The learning slope can be estimated using techniques described in Dar-El (2000). We assume that the work elements are small in comparison to the cycle time so that each station is assigned over seven tasks. Under this assumption, moving a task to another station or swapping a pair of tasks between stations, have only marginal effect on the overall station's learning slope (even if the tasks have widely different slopes).

Thus, a station *i* can be characterized using the following parameters:

STD$_i$ -The standard time of a station *i*, is the total standard time of its allocated tasks.

std$_j$ - The standard time of a task *j* ,

Defining *k* as the number of tasks allocated to station *i*, we have:

$$STD_i = \sum_{j=1}^{k} STD_j \qquad (7)$$

t$_{i,1}$ - The first cycle time of station *i*, is the total time for the first cycles of its allocated tasks. The first cycle of the allocated tasks should be computed from their standard time and learning slope using equation 5. For the calculation of the stations cycle time we shall use the following definitions:

tt$_{j,1}$ = The first cycle time of a task *j*.

k$_i$ = the number of tasks allocated to station *i*

Thus**,** we have:

$$\text{First cycle-time at station } i = t_{i,1} = \sum_{j=1}^{k_i} tt_{j,1} = \sum_{j=1}^{k_i} \left(std_j \cdot G(\phi)_j \right) = \sum_{j=1}^{k_i} \left(std_j \cdot (57 - 60 \cdot \phi_j) \right) \qquad (8)$$

Thus, the first cycle of each station could be computed by knowing the standard times and the learning slopes of the tasks at each station can generate

For computing the weighted average for a station's learning slope, we need to develop the weights. The weights reflect the total workload induced by the task throughout the learning process. We use the term $LOAD_j$ as the total work load contributed by task *j* over all *M* cycles in the lot. Utilizing equation 4, we get:

$$LOAD_j = \left(\frac{tt_j}{(1 - b_j)} \right) \cdot \left(M^{(1 - b_j)} \right) \qquad (9)$$

b$_i$ - Stations' *i* learning constant is computed as a weighted average of its learning constants. We chose LOADj as weights, since they are aggregate reflection of all the cycles' contribution to the workload. Thus, defining the tasks learning constants ***tb$_j$,*** and the number of tasks allocated to station *i* to be ***k***, ***b$_i$*** is computed as follows:

$$b_i = \frac{\sum_{j=1}^{k} \left(tb_j \cdot LOAD_j \right)}{\sum_{j=1}^{k} LOAD_j} \qquad (10)$$

THE OPTIMIZATION FRAMEWORK AND DATA PREPARATION

The proposed optimization is based on two main stages. In the first stage we relax the non-divisibility constraints of tasks and assume that a task could be split between any pair of adjacent stations. We assume that the work elements are all small in comparison to the cycle time (as is the case for low volume demand) so that each station contains several tasks. Under this assumption, the stations' learning slopes of a base allocation of tasks to stations would not be sensitive to moving a task to another station or swapping a pair of tasks between stations. These actions would have only marginal effect on the overall station's learning slope (even if the tasks have widely different slopes). With the above relaxation, assumption, and rough estimates of the learning slopes of all the stations, we can replace the actual task assignment to stations with general workload assignment to stations.

Minimizing the makespan can be translated into maximizing the simultaneous work of all stations and this is achieved when the learning slopes are equal (Dar-El 2000). Cohen et. al. (2006) showed that in this special case of equal slopes, the optimal work allocation to stations should not be equal. In most cases, however, precedence constraints would prevent us from achieving this ideal identical slope. These cases are characterized by relatively low flexibility of the precedence network of assembly activities. For example, using the flexibility gauge called F-ratio (defined by Dar-El (1973)) Bukchin and Rubinovitz (2003) report on typical F-ratio values of 0.2 and 0.3 (low flexibility) for assembly precedence diagrams. Thus, one should expect that the precedence constraints would dictate to a significant extent the possible work content of the various stations and consequently their learning slopes.

Prior to the optimization, the following preparations are needed:

(i) Analyze the project, breaking it down to indivisible tasks and build the precedence diagram.

(ii) Use a PMTS technique (Predetermined Motion Time Standards) to determine the standard time for each task.

(iii) Determine the learning slope for each task and as a result compute $G(\phi)=57-60\phi$ (see equations 5 & 6).

(iv) Make a rough cut estimation of which tasks could be assigned to which station using any line balancing technique (for more on this issue see section 6).

(v) Compute the standard time for each station using equation 7.

(vi) Compute the learning constant for each station using equation 10.

(vii) Sort the station numbers by decreasing learning constant (b_j). For example, the sorted list for three stations with $b_1<b_2<b_3$, is {3, 2, 1}. This sorted list is ordered according to the segments in the optimal upper envelope (Theorem 1).

(viii) Use the list from (vii) to identify the intersection points of each pair of learning curves in the upper envelope (all adjacent pairs in the list of (vii)). Use the notation $Q_{i,j}$ for the intersection point between stations i and j. e.g., For the example of three stations with $b_1<b_2<b_3$, the intersections would be: $Q_{3,2}$ and $Q_{2,1}$.

Once these preparations are completed, begin the optimization procedure which includes steps (A) and (B) as follows:

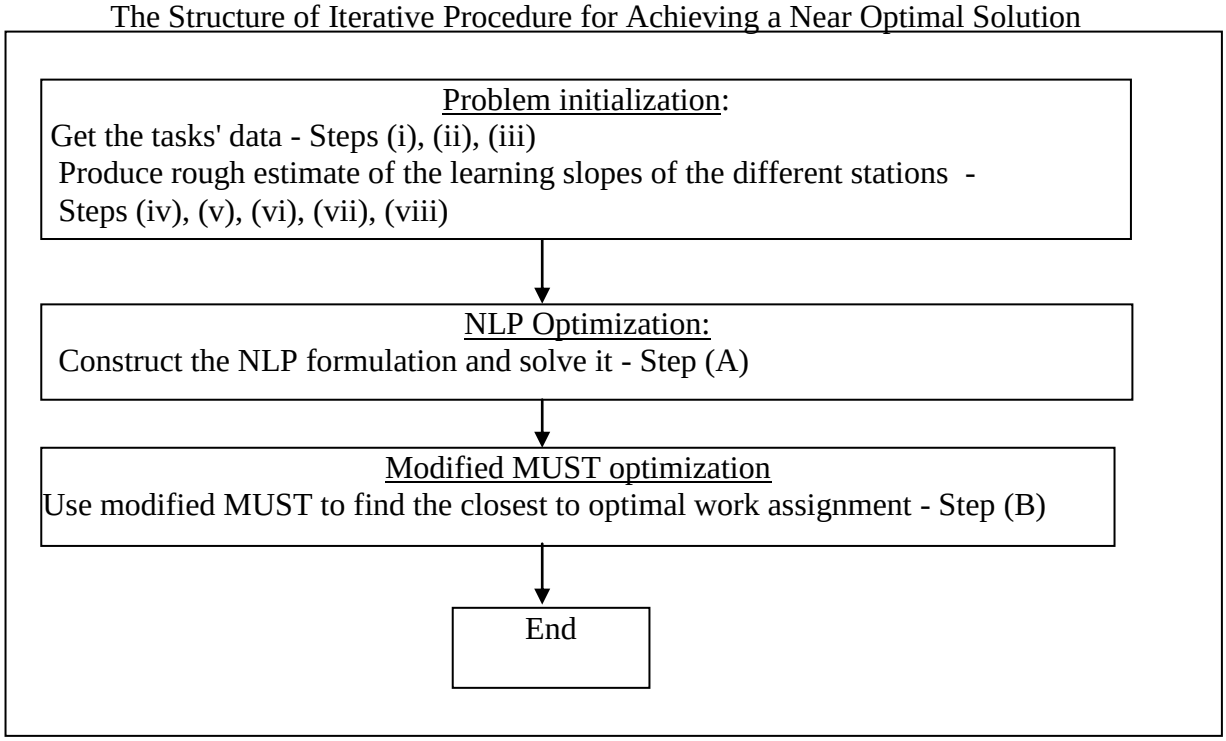
A. Formulate the problem as a Non Linear Programming (NLP) problem and solve it to determine the optimal workload allocation for the relaxed problem (assuming tasks can be split between adjacent stations).
B. Drop the tasks divisibility relaxation and use a modified MUST technique to find the non-relaxed work assignment closest to the relaxed optimal solution found in (A)

Step (B), while dropping the relaxations, is based upon the following important assumption:

There are several tasks per station (say, over 5), so that swapping a pair of work elements between adjacent stations or moving a single element will not change drastically the stations' learning slopes.

Figure 3 summarizes the iterative procedure:

FIGURE 3
The Structure of Iterative Procedure for Achieving a Near Optimal Solution



THE NONLINEAR OPTIMIZATION FORMULATION

The following notation is used throughout the formulation

Notation:

L - Number of stations in the line.

M - The total number of products to be produced.

STD - The total standard time to produce one product according to time study analysis.

$\boldsymbol{\phi}$ - The learning slope (the reduction in cycle time when the quantity produced is doubled.)

$\mathbf{G(\phi)_i} = (57\text{-}60\phi_i)$ - The ratio between the i^{th} station's first cycle time and the standard time allocated to that station. This ratio depends only on the standard work assigned and the learning slope of the station.

$\boldsymbol{b_i}$ - learning constant of the i^{th} station along the line ($b_i = -\text{Log}_2\phi_i = -\text{Log}(\phi_i)/\text{Log}2$).

$\boldsymbol{ORD_0}$ - List of station numbers sorted by a descending order of learning constants. Stations are uniquely identified by their sequential location along the line. For example, if b_1=0.17, b_2=0.2, and b_3=0.25, then ORD_0={3, 2, 1} whereas the stations along the line are ordered: 1→2→3.

$\boldsymbol{ORD_{(i)}}$ - The ith entry in ORD_0. For example, $ORD_{(1)}$= 3 gives the number of the station that has the largest learning constant *b*. (In the above example $ORD_{(1)}$=3, $ORD_{(2)}$=2 and $ORD_{(3)}$=1.)

$\boldsymbol{b[ORD_{(i)}]}$ - learning constant of the i^{th} station in ORD_0. (For example, $b[ORD_{(1)}]$ is the largest learning constant, and $\boldsymbol{b[ORD_{(L)}]}$ is the smallest constant among the stations learning constants.)

$\boldsymbol{Q_{(ORD(i),ORD(j))}}$ - The cycle in which the cycle times of two bottleneck stations are equal in both stations ($ORD_{(i)}$,$ORD_{(j)}$: i,j∈1, 2,…, *L*). Each intersection point is associated with two segments on the upper envelope. We shall also use the convention that: $Q_{0,1} = 1$, $Q_{L,L+1} = M$.

$\boldsymbol{t_{(ORD(i),ORD(,j))}}$ - The execution time for the j^{th} cycle of the i^{th} station in the sorted list - ORD_0).

$\boldsymbol{t_{ORD(i),1}}$ - The first execution time for the ith station in ORD_0.

W - The MAKESPAN or the total assembly time for "M" units.
$\boldsymbol{D_{i,j}} = ORD_{(j)} - ORD_{(i)}$ = The sequential difference along the line between stations $ORD_{(i)}$ and $ORD_{(j)}$.

The nonlinear programming (NLP) formulation

The objective function is to minimize the makespan which is the sum of the areas below the segments of the upper envelope. The objective function is expressed by all the variables the intersection points as well as the first cycle times.

Objective function**:**

Minimize W (where W=makespan is expressed as area beneath the upper envelope):

$$W = \sum_{k=1}^{(ORD_{(1)}-1)} t_{k,1} \qquad (12)$$

(The summation of the first cycles until the first bottleneck segment)

$$+\sum_{i=1}^{L}\left(\left[\frac{t_{ORD_{(i)},1}}{1-b_{ORD_{(i)}}}\right]\left[\left[Q_{(ORD_{(i)},ORD_{(i+1)})}\right]^{(1-b_{ORD_{(i)}})} - \left(Q_{ORD(i-1),ORD(i)}\right)^{\left(1-b_{ORD_{(i)}}\right)}\right]\right)$$

(Summation of the area beneath the upper envelope (under the various segments)).

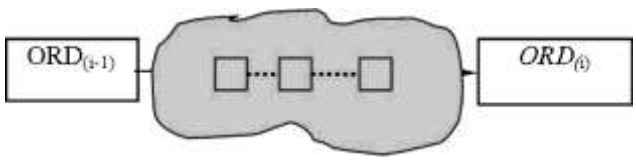
$$+\sum_{k=ORD_{(L)}+1}^{L}\left(t_{k,1}\cdot M^{-b_k}\right)$$ (sum of last cycles of the stations that follow the last bottleneck)

Subject to:

$$STD = \sum_{i=1}^{L}\left(\frac{t_{i,1}}{G(\phi)_i}\right) \qquad (13)$$

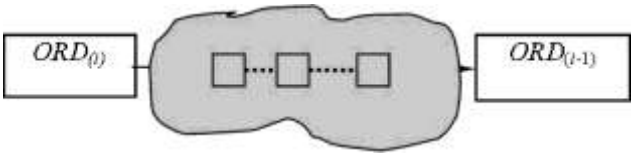
The total standard time - *STD* - is related to the sum of first cycle times divided by G(ϕ). The sum of the Standard Times (from Work Measurement techniques) at all stations is fixed and assumed to be known.

Equations 14 and 15 define the intersection points of pairs of learning curves on the upper envelope. These point mark the beginning and end of each upper envelope segment. The order of segments of the optimal upper envelope is given in $ORD_{()}$. Therefore, for every adjacent pair of stations in the *ORD* list write either equation 14 or 15 according to the case:
If station $ORD_{(i\text{-}1)}$ location along the line is before station $ORD_{(i)}$ - then:



$$t_{[ORD(i),1]}\cdot\left(Q_{[ORD(i-1),ORD(i)]}^{\left(-b_{[ORD(i)]}\right)}\right) = t_{[ORD_{(i-1),1}]}\cdot\left(Q_{[ORD(i-1),ORD(i)]} + D_{[(i-1),i]}\right)^{\left(-b_{[ORD(i-1)]}\right)} \qquad (14)$$

Else If station $ORD_{(i)}$ location along the line is before station $ORD_{(i\text{-}1)}$ - then:



$$t_{[ORD(i),1]} \cdot \left(Q_{[ORD(i-1),ORD(i)]}^{\left(-b_{[ORD(i)]}\right)} + D_{[(i-1),i]} \right) = t_{[ORD_{(i-1)},1]} \cdot \left(Q_{[ORD(i-1),ORD(i)]} \right)^{\left(-b_{[ORD(i-1)]}\right)} \quad (15)$$

There are L-l of either 14 or 15 constraints, (since i can have only the following values: i=L, L-1,...,3,2).)

$$Q_{[ORD(i-l),ORD(i)]} + \text{Max}\{ORD_{(i-1)},ORD_{(i)}\}\text{-}1 < Q_{[ORD(i),\ ORD(i+1)]} + \text{Max}\{ORD_{(i)},ORD_{(i+1)}\}\text{-}1 \quad \forall$$

$$i=1,2,...,L \quad (16)$$

Type (16) inequalities express the optimal order of the segments in the upper envelope, through the use of the breakeven cycles. All other elements in the inequality are related to the phase difference between the pair of stations of each breakeven point (e.g., the cycle number in station $ORD_{(i)}$ vs. the cycle number in $ORD_{(i-1)}$ at the two cycles breakeven time). There are L type (16) inequality constraints (i = 1,2,...L}. Note that from theorems 1 and 2, all $ORD_{(i)}$ are known constants and thus, there are only two variables in each equation of type (16), namely: $Q_{[ORD(i-l),ORD(i)]}$ and $Q_{[ORD(i),\ ORD(i+l)]}$.

Discussion of the NLP formulation

The first summation in (12) $\sum_{k=1}^{(ORD_1-1)} t_{k,1}$, expresses the time it takes the first product to reach the first bottle neck station. The second summation in (12): $+\sum_{i=1}^{L}\left(\left[\frac{t_{ORD_i,1}}{1-b_{ORD_i}}\right]\left[\left[Q_{(ORD_i,ORD_{i+1})}\right]^{(1-b_{ORD_i})} - \left(Q_{ORDi-1,ORDi}\right)^{(1-b_{ORD_i})}\right]\right)$ describes the continuous upper envelope. i.e. the duration of bottleneck stations. The third summations in (12): $+\sum_{k=ORD_L+1}^{L}\left(t_{k,1}\cdot M^{-b_k}\right)$, are the times of the last product on stations located after the last bottleneck station. Since the segments of the optimal upper envelope are sequenced by decreasing learning slope, the last bottleneck is the station with smallest learning slope. The last finished product on this station has to be processed on the rest of the machines until it is ready. This additional time is expressed in the third summation.

Constraint (13): $STD = \sum_{i=1}^{L}\left(\frac{t_{i,1}}{G(\phi)_i}\right)$, merely defines that total standard time to be the sum of all stations standard times. Constraints 14 and 15 define the intersection points of every two curves on the upper envelope.

To understand the role of the $D_{i,j}$, remember that the first station always processes one cycle ahead of the second station, two cycles ahead of the third station, and so on. With the effects of blockage and starvation at any station i, i is j cycles ahead of station i+j. The breakeven cycle $Q_{ORDi,ORDj}$ always refers to the further <u>downstream</u> station - Max{$_{ORD(i)}$, $_{ORD(j)}$} i.e the station with the <u>smaller</u> number of cycles. Thus, $D_{i,j}$ reflects the fact that at the time the first product arrives at station j, station i already had an experience of $D_{i,j}$ =ORD_j-ORD_i cycles.

The order of curves' segments in the upper envelope is determined by decreasing learning constants ($\boldsymbol{b}_{ORD(i)}$) (this is a must for avoiding total dominance which indicates non-optimality). Therefore, the identity and order of the two stations forming each intersection point on the optimal upper envelope are also known. This knowledge is expressed in (16) by specifying the order the break-even cycles ($Q_{i,j}$). Since the upper envelope describes the production of the whole line, the intersections cycles ($Q_{i,j}$) must be expressed in terms of the whole line rather then the three stations involved ($ORD_{(i-1)}$, $ORD_{(i)}$, $ORD_{(i+1)}$) in the two break-even points: (1)$Q_{ORD(i-1),ORD(i)}$, *(2)* $Q_{ORD(i),ORD(i+1)}$. This is done by taking station number 1 in the line as a point of reference for counting cycles. Since ($Q_{ORD(i),ORD(i-1)}$) expresses the number of cycles at the further downstream station. [Max{ORD_{i-1}, ORD_i}-1], the number of cycles the first product had done before arriving at that downstream station is added to the left side of (16). By the same token, ($Q_{ORD(i),ORD(i+1)}$) expresses the number of cycles at the further downstream station. [Max{$ORD_{(i)}$, $ORD_{(i+1)}$}-1], the number of cycles the first product had done before arriving at that downstream station is added to the right side of (16). Thus, the reference to the first station is established.

Discussion of the NLP optimization results

Experimenting with hundreds of different configurations several general trends were detected. Table 1 summarizes the influence of three factors on the first cycle times in the various stations. These first cycles are indicative of the work allocation in the optimal solution. The three factors are: the learning slope b_i, the total quantity produced M, and the number of stations L. The results of four cases in Table 2 are representative and supports these general conclusions.

TABLE 1

Factors Affecting Work Allocation to Stations with Varying Learning Slopes

Factor	Its Effect on optimal $t_{i,1}$ (first cycle time of station i)
Learning Slope - b	When b_i is larger (higher learning rate) the optimal $t_{i,1}$ tends to be larger, and when b_i is smaller $t_{i,1}$ tend to be smaller.
Quantity produced - M	As M increases the optimal allocation of work converges to an equal allocation of standard time to all stations. However, diversity of the $t_{i,1}$ remains due to differences in G(ϕ).
Number of Stations - L	Increases in L reduces the diversity of $t_{i,1}$ (and naturally decreases all $t_{i,1}$ values)

Table 2 considers four cases of three stations lines. In all cases the total assembly standard time is 20 time units. The stations were tested for M=100 and M=900 products, with monotonically increasing and decreasing learning slopes. The two configurations of the learning slopes are: [70%, 80% 90%]. and [90%, 80%, 70%]. The aim is to allocate this work to the stations in an optimal way.

As mentioned in the introduction, balanced lines (or, equal work allocation) is the method generally used in assembly line design. We therefore compare the makespan generated using the optimal allocation method with those obtained for equal station work allocations. In all cases, the optimal allocation brings a significant reduction in makespan time. Table 2 compares the the makespan of the four cases. In every case, the optimal allocation makespan is shorter than that obtained from equal station allocations. Indeed, the

percentage reductions ranging from 9% to over 17% must be considered as being highly significant.

TABLE 2
Optimal Work Allocation for Four Cases in a Three Stations Line

Batch Qty.	**Order of learning slopes (%)**	**Optimal makespan**	**Makespan of equal work allocation to stations**	**Percent savings**	**First Station Std. Cycle time**	**| 1st time**	**Second Station Std. Cycle time**	**| 1st time**	**Third Station Std. Cycle time**	**| 1st time**
M	$\phi_1 \rightarrow \phi_2 \rightarrow \phi_3$	**Min. Fmax**		**(% savings)**	STD_1	$t_{1,1}$	STD_2	$t_{2,1}$	STD_3	$t_{3,1}$
100	70→80→90	1,957.0	2,261.7	**15.5**	5.44	81.0	5.59	50.4	8.96	26.9
100	90→80→70	1,956.5	2,301.6	**17.6**	9.15	27.4	5.66	51.0	5.18	77.7
900	70→80→90	8,418.5	9,213.4	**9.4**	8.99	134.9	5.40	48.6	5.59	16.8
900	90→80→70	8,453.6	9,227.7	**9.1**	5.60	16.8	5.40	48.6	8.98	134.8

Note, that for a given production quantity, the optimal makespans are highly similar for ascending and descending order of learning slopes. Based on extensive simulation runs, we find that the savings from using optimal work allocations are sensitive to the following factors: the learning slope, the number of stations and the total quantity produced. These effects are summarized in Table 3.

TABLE 3
Factors affecting the Makespan Savings - Optimal .vs. Equal - Work Allocation

The Factor	**Its effect on the makespan savings in comparison to equal work allocation**
Learning Constant (***b***)	Increased savings with wider diversity in the value of *b*
Quantity Produced (***M***)	Increasing *M* automatically increases the makespan, and therefore the absolute savings, but the percentage of savings out of the increased makespan drops.
Number of stations (***L***)	Increased savings with increasing number of stations, *L*

FINDING THE NEAR OPTIMAL SOLUTIONS USING MUST

After finding the NLP ideal solution which ignores the indivisibility of tasks, we now find the closest feasible solution to that ideal solution. We modified the MUST (Dar-El and Rubinovitz 1979) line balancing technique to generate all the possible task assignments solutions within 5% of the optimal first cycle times and learning slopes (within 5% of each station NLP targets). Unlike other optimal line balancing methods which give a single optimal solution to the assembly line balancing problem (ALBP), MUST finds all the optimal solutions for the problem. MUST was traditionally used in line balancing to allocate tasks to stations with identical cycle time target for all stations, but now MUST must allocate the first

cycle times of the work elements, to the stations' optimal first cycle times obtained by the NLP. MUST is based on a very efficient enumerative algorithm that gives it great flexibility and the possibility to adjust for different cycle times in different stations. It also has the flexibility to keep the solutions according to certain criteria.

Thus, the modified MUST takes as an input: (1) all the first cycle times of all the work elements to be allocated to stations, (2) the learning slopes of each work element, (3) the optimal first cycle of each station from the NLP (4) the NLP learning slope of each station.

The modified MUST is instructed to fill the stations with up to 105% of their NLP first cycles. Solutions that have stations a first cycle below 95% of their target are dropped. Also, the stations' learning slopes are computed according to equation (10) and solutions with deviations of 0.05 or more from their learning slopes are dropped. If no feasible solution was found by MUST, a new run is made, but this time with allowable deviation of 7%. (MUST is instructed to fill the stations with up to 107% of their NLP first cycles and to drop those with a first cycle below 93% of their target.) This procedure is repeated iteratively, by increasing the allowable deviation from the first cycle times of the NLP solution, until a feasible solution has been found by MUST.

In all the runs that were tested, the makespan of the selected MUST solution (having different work allocations to different stations) was much shorter than the makespan of the perfectly balanced allocation (i.e., equal work allocation to stations). A simple example in section 6 illustrates the proposed method and presents the makespan savings when compared to an equal work allocation to stations.

A SIMPLE ILLUSTRATION

Consider the following simple example, with 3 stations, and a lot of 100 assemblies, each with 24 activities. The example uses generic time units throughout.

Legend:

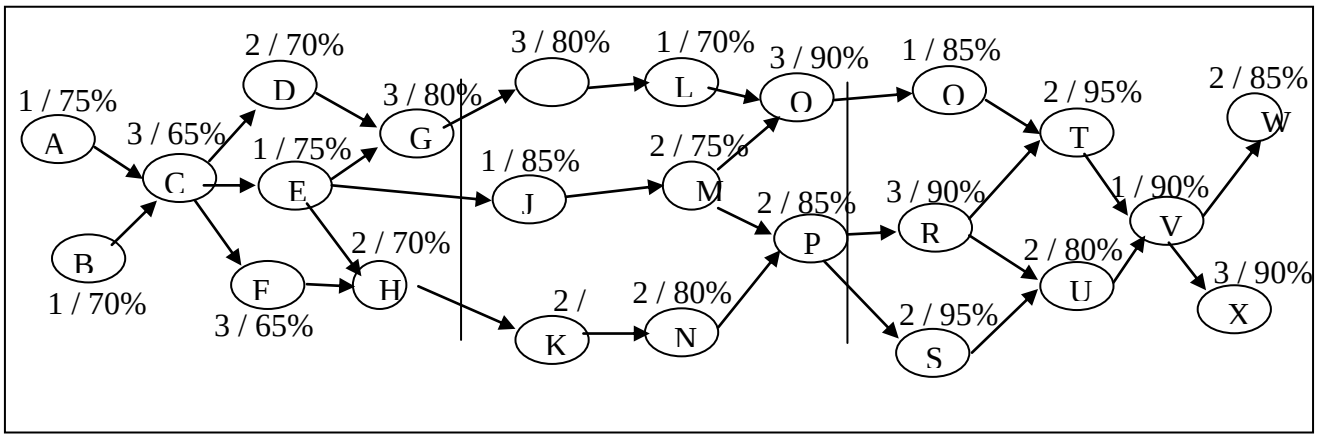
A - An activity node (node names are alphabetically ordered by precedence constraints)

→ - Precedence relation

2 / 80%- Standard time / Learning slope

FIGURE 4

An Example Network for the Illustration of Fitting the Best Solution



Applying a simple balancing algorithm by Kilbridge and Wester (1961) yields a theoretical perfect balance of standard times with the following allocation (the computed standard time, learning slope and first cycle are given for each station):
Station 1: A, B, C, D, E, F, G, H → $STD_1 = 16$, $\phi_1 \cong 70\%$, $t_{1,1}=240$.
Station 2: I, J, K, L, M, N, O, P → $STD_2 = 16$, $\phi_2 \cong 80\%$, $t_{2,1}=144$.
Station 3: Q, R, S, T, U, V, W, X → $STD_3 = 16$, $\phi_3 \cong 90\%$, $t_{3,1}=48$.

This or an equivalent solution could be obtained using almost any line balancing technique. The only important consideration in this stage is to get an estimation of the stations' learning slopes.
The learning curves and the upper envelope of this solution are depicted in Figure 5(a).

Running the NLP for three stations with these learning slopes gives:
Station 1: Optimal standard time = $STD_1^* = 13$, ($\phi \cong 70\%$) , $t_{1,1}^*=195$.
Station 2: Optimal standard time = $STD_2^* = 14$, ($\phi \cong 80\%$), $t_{2,1}^*=126$.
Station 3: Optimal standard time = $STD_3^* = 21$, ($\phi \cong 90\%$), $t_{3,1}^*=63$.

FIGURE 5 (a)
The Learning Curves of the Equal Standard Time Allocation for the Three Station of the Example

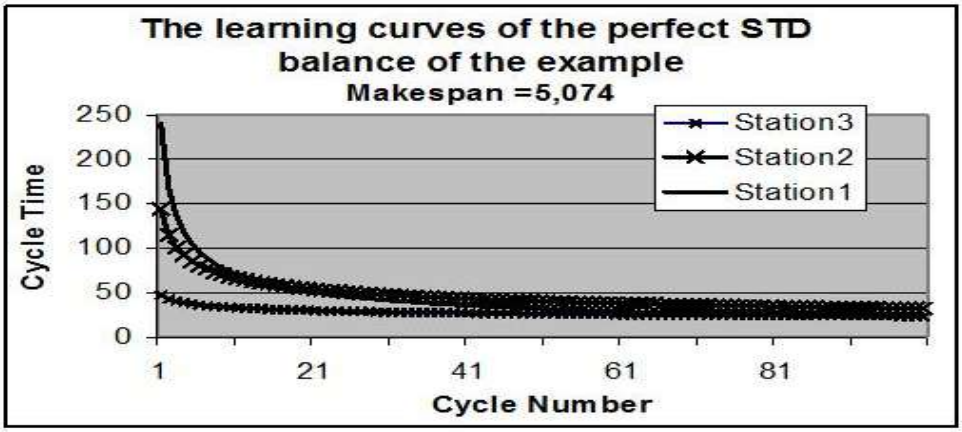
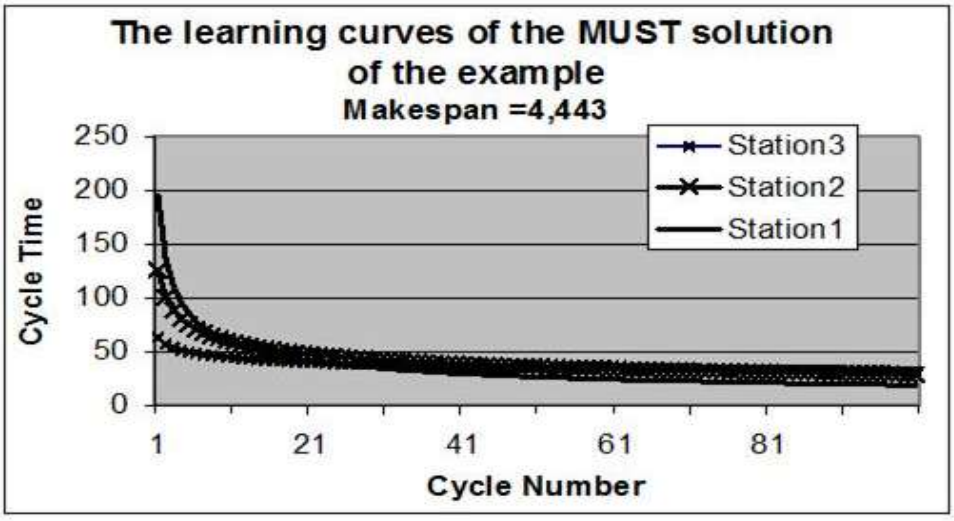


FIGURE 5 (B)
The Learning Curves of the MUST Solution for the Three Station of the Example



Running MUST on the above network with the NLP targets, we find several solutions from which we select the closest in terms of learning slope:

Station 1: A, B, C, D, E, F, H → STD_1= 13, Estimated learning slope ≅ 70%, $t_{1,1}$=195.

Station 2: G, I, J, K, L, M, N, → STD_2= 14, Estimated learning slope ≅ 80%, $t_{2,1}$=126.

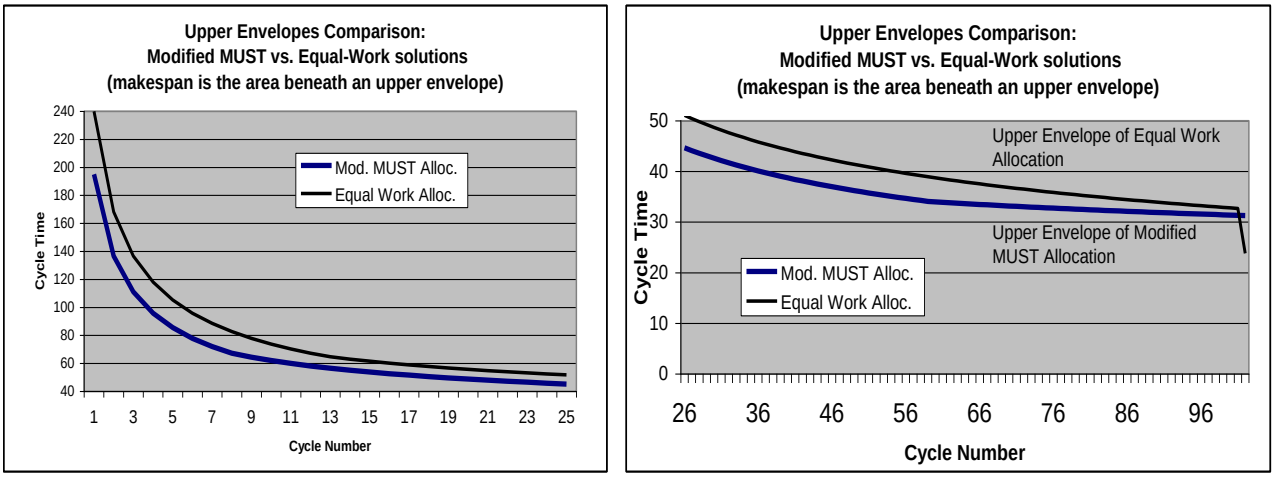
Station 3: O, P, Q, R, S, T, U, V, W, X → STD_3= 21, Estimated learning slope ≅ 90%, t_1=63.

The learning curves of this solution are depicted in Figure 5(b) and the upper envelope in figure 6.

Thus, by using the proposed approach (combining NLP and modified MUST) the makespan was reduced from 5,074 to 4,443 time units resulting in makespan savings of 12.4% (or 631 time units).

FIGURE 6

Comparison of the Upper Envelopes of the Equal Work Allocation vs. the Modified MUST Work Allocation (The Makespan is the Area under the Upper Envelope).



Improving the projection quality of the a-priori stations' learning slopes

While the above example illustrates the principles of the proposed optimization, it also highlights the possible contribution of a better rough cut a-priori estimation of which tasks could be assigned to which station. A better estimation would be closer to the final work allocation than the allocation of a regular line balancing. We therefore propose 2 ways for improving this a-priori estimation:

1. Modify any line balancing technique by replacing the activities standard times by the times of the half lot cycle (M/2).
2. Balance the line based on the task load ($Load_j$ - see equation 9). This could be accomplished by using any line balancing technique with two replacements:

I. Replace the task time by $Load_j$,

II. Replace the station cycle time by: $\frac{1}{L}\sum_{j=1}^{last\_task} Load_j$ (11)

A rough estimate of the learning slope of each station could be calculated using the solution of modified line balancing.

CONCLUSIONS

In this paper we discuss the characteristics of minimizing the makespan of a production batch with variable task learning performed on an assembly line without buffer

space between consecutive stations. We considered the possibility that the various stations may have different learning slopes. We introduced a two stage solution technique: (1) the NLP optimization of the relaxed problem, and (2) the modified MUST technique. The NLP optimization concept and factors affecting its savings (in relation to equal/balanced work allocation) are discussed and analyzed. A simple example was given to illustrate the usage of the modified MUST. The case where buffer spaces exist between stations is a subject for further research. This study shows that for moderate quantities of total product demand, significant savings in makespan can be

REFERENCES

Adler, G. L. and Nanada, R. (1974). The effect of learning on optimal lot size determination - the simple product case. *AIIE Transactions, 6,* 14-20.

Ardity, D., Tokdemir O.B., and Suh K. (2001). Effect of learning on line of balance scheduling. *International Journal of Project Management, 19,* 265-277.

Bukchin Y., Rubinovitz Y. (2003). A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions, 35,* 73-85.

Chakravarty, A. K. (1988). Line balancing with task learning effects. *IIE Transactions, 20,* 186-193.

Cohen, Y. and Dar-El, E. M. (1998). Optimizing the number of stations in assembly lines under learning. *Production Planning and Control, 9,* 230-240.

Cohen, Y., Vitner G., and Sarin S. (2006). Optimal Allocation of Work in Assembly Lines for Lots with Homogenous Learning. *European Journal of Operational Research (Special issue on line balancing) 168* (3), 922-931.

Cohen, Y., Vitner G., and Sarin S., 2008. Work Allocation to stations with varying learning slopes and without buffers, European Journal of Operational Research, 184 (1), 797-801.

Dar-El, E. M. (1973). MALB - a heuristic technique for balancing large single model assembly lines. *AIIE Transactions, 5* (4), 343-356.

Dar-El, E. M., and Rubinovitz, Y. (1979). MUST - A Multiple Solutions Technique for Balancing Single Model Assembly Lines. *Management Science, 25* (11), 1005-1114.

Dar-El, E. M., and Rabinovitch, M. (1988), Optimal planning and scheduling of assembly lines. *International Journal of Production Research, 26,* 1433-1450.

Dar-El, E. M., and Mazar, A. (1989). Predicting the performance of unpaced assembly lines where one station variability may be smaller than the others. *International Journal of Production Research, 27,* 2105-2116.

Dar-El, E. M., Ayas, K., Gilad, I. (1995a). A dual-phase model for the individual learning process in industrial tasks. *IIE Transactions, 27,* 265-271.

Dar-El, E. M., Ayas, K. and Gilad, I. (1995b). Predicting performance times for long cycle time tasks. *IIE Transactions, 27,* 272-281.

Dar-El, E. M. (2000). *Human Learning: From Learning Curves to Learning Organizations*. Boston: Kluwer Academic Publishers.

Erel E., Sarin S. C. (1998). Survey of assembly line balancing procedures. *Production Planning and Control, 9* (5), 414-434.

Gilad, I., Dar-El, E. M., and Brenner, M. (1991). Learning curve parameter prediction. *Proceedings of the XIth International Conference of Production research*, (Hefei, China) 602-612.

Globerson, S., and Shtub, A. (1984). The impact of learning curves on the design of long cycle time lines. *Industrial Management, 26,* 5-10.

Kilbridge, M. D., and Wester, L. (1961). A heuristic method of assembly line balancing. *Journal of Industrial Engineering, 12,* 292-298.

Kroll, D. E. (1989). The incorporation of learning in production planning models. *Annals of Operations Research, 17,* 291-304.

Yelle ,L. E. (1979). The learning curve: historical review and comprehensive survey. *Decision Science, 10,* 302-328.