ASSEMBLY LINE BALANCING:
A BRANCH AND BOUND APPROACH

by

MUSTAFA BÜYÜKABACI

B.S. in I.E., Boğaziçi University, 1984

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

Boğaziçi University

1986

ASSEMBLY LINE BALANCING:

A BRANCH AND BOUND METHOD

APPROVED BY

Doç. Dr. Gündüz ULUSOY (Thesis Supervisor) .....

Doç. Dr. Ceyhan UYAR .....

Yar. Doç. Dr. Mahmut KARAYEL .....

DATE OF APPROVAL Jan. 30, 1987

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis supervisor Doç. Dr. Gündüz ULUSOY for his guidance during this study.

I would like to express my sincere thanks to Yard. Doç. Dr. Mahmut KARAYEL and Doç. Dr. Ceyhan UYAR for serving on my thesis commitee.

I am also indebted to Doç. Dr. M. Akif EYLER for his suggestions during the programming phase.

Finally, I should like to mention the help and contributions of my friends, O. Nurettin UÇ and Enes YILDIZ, during the evaluation and documentation phases.

ABSTRACT

Assembly Line Balancing (ALB) problem is a combinatorial problem which has extremely large number of feasible solutions. In the literature, there are two approaches to the solution of this problem: heuristic methods and exact methods.

Heuristic methods give suboptimal solutions but they are easy to compute. Whereas, exact methods give the optimum solutions but they possess severe computational difficulties for sufficiently large problems in real-life. Therefore, exact methods have to be improved to solve real-life ALB problems with less difficulty.

In this study, it is intended to obtain optimal solutions in real-life ALB problems that would be effectively used in industry. Therefore, a package of user-friendly interactive computer programs is developed in which a new branch and bound (B&B) approach is implemented.

The method is studied for single-model and mixed-model assembly lines. In order to make the problem more realistic, additional restrictions are added to the regular precedence and cycle time restrictions. An upper bound concept which is used for feasibility check of the lower bounds is introduced.

The programs are run for several sample problems and computational efficiency of the B&B method proposed is obtained.

ÖZET

Montaj Hattı Dengelemesi (MHD) problemi cok sayıda uygun çözümlere sahip olan kombinatoryal bir problemdir. Literatürde bu probleme iki değişik şekilde yaklaşılmıştır: sezgisel metodlar ve kesin çözüm metodları.

Sezgisel metodlar genellikle optimum sonuç vermezler fakat kullanımları kolaydır. Diğer taraftan kesin çözüm metodları optimum sonuç verirler fakat gerçek hayatta karsılasılan buyuk problemlerde ciddi hesaplama zorluklarına sahiptirler. Bu yüzden, kesin çözüm metodları gerçek hayattaki MHD problemlerini daha az bir zorlukla çözebilecek şekilde geliştirilmek zorundadırlar.

Bu çalışmada, gerçek hayattaki MHD problemlerine endüstride etkili bir şekilde kullanılabilecek optimal çözümler getirmek amaçlanmıstır. Bu yüzden, yeni bir dal-sınır metodu (D-S) yaklaşımının uygulandığı etkileşimli bilgisayar programları paketi geliştirilmistir.

Bu metodda, tek modelin ve karışık modellerin üretildiği hatlarda çalışılmıstır. Problemi daha gerçekçi yapmak için öncelik/sonralik ve çevirim zamanı kısıtlarına ilave olarak bazı özel kısıtlarda eklenmiştir. Alt sınırların uygunluğunun kontrolünde kullanılan bir üst sınır kavramı ilave edilmiştir.

Bilgisayar programları çesitli örnek problemler için denenmiş ve ortaya konulan bu yeni D-S metodunun hesaplama verimliliği gözlenmiştir.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b : Maximum proportion of cycle time to be utilized

c_j : Penalty associated with using workstation j, $j \in J$.

I : the task set , I = {1,2,...,n}

J : the station set, J = {1,2,...,m}, where $m \leq n$

LB : Lower bound

M_j : Average work content at station j

m : Number of workstations

N_p : Least number of stations for unassignedoperations

N_r : Current station level, number of stations for assigned operations

P : Penalty

P(i) : { Immediate predecessors of task i }

p_j : Total work content of station j

q : Minimum probability for the station time not exceeding the cycle time

S_i : Orientation of task i

T : Sum of assigned operation times

T_c : Cycle time

t_i : The process time of task i , $i \in I$

T_w : Total work content

TIT : Total idle time

U_1 : Strict grouping of operations

U_2 : Same station elements

U_3 : Prefered grouping of operations

U_4 : Strictly different station elements

U_5 : Complementary operations

$$x_{ij} = \begin{cases} 1 & \text{, if task } i \text{ is assigned to station } j, \\ & j \in J,\ i \in I. \\ 0 & \text{, otherwise} \end{cases}$$

I. INTRODUCTION

1.1. A Historical Perspective

The increasing need for finished goods in large quantities has, in the past, led engineers to search for and to develop new methods of production. Many individual developments in the various branches of manufacturing technology have been made and have allowed the increased production of improved finished goods at lower costs. One such important development has been the introduction of the assembly process. This process is required when two or more parts and/or components are to be brought together to produce a component or the finished product.

Boothroyd, Poli and Murch [1] give the following historical account of the development of the assembly lines and their management. The early history of assembly process development is closely related to the history of the development of mass-production methods. Thus the pioneers of mass-production are also the pioneers of modern assembly process. Their new ideas and concepts have brought significant improvements in the assembly methods employed in large-volume production.

In the early stages of manufacturing technology, the complete assembly of a product was carried out by a single operator and usually this operator also manufactured the

individual component parts of the assembly. Consequently, it was necessary for the operator to be an expert in all the various aspects of the work, and training a new operator was a long and expensive task. The scale of production was often limited by the availibilty of trained operators rather than by the demand for the product.

Eli Whitney, in United States in 1798, made an attempt on mass-production. He offered to contract to manufacture a large number of muskets, when the federal arsenals could not meet the demand in a short period of time. Although he could not achieve to complete the contract, his ideas on mass-production had been successfully proven. The factory was built specially for the manufacture of muskets, contained machines for producing interchangable parts. These machines reduced the skill required by the various operators and allowed significant increase in the rate of production.

The results of Eli Whitney's work brought about three primary developments in manufacturing methods:

(i) Parts were manufactured on machines, resulting in a consistently higher quality than that of hand-made parts. These parts were now interchangable and as a consequence assembly work was simplified.

(ii) The accuracy of final product could be maintained at a higher standard.

(iii) Production rates could be significantly increased.

Conception of conveying materials from one place to another without manual effort, by Oliver Evans in 1793, led

eventually to further developments in automation assembly. The next significant contribution to the development of assembly methods was made by Elihu Root in 1849. He divided the operations of assembling the component parts into basic units that could be completed more quickly and with less chance of error. This gave rise to the following concept: "Divide the work and multiply the output".

Frederick Winslow Taylor introduced the methods of time and motion study in 1881. The objective was to save the operator's time and energy by making sure that the work and all the things associated weith the work were placed in best positions for carrying out the required tasks.

Undoubtedly, the principal contributor to the development of production and assembly methods was Henry Ford. He described his principles of assembly in following words:

(i) Place the tools and then men in the sequence, so that each part will travel the least distance.

(ii) Use some form of carrier so that when a workman completes his opeartion, he drops the part always in the same place.

(iii) Use sliding assembly lines by which parts to be assembled are delivered at convenient intervals, spaced to make it easier to work on them.

Ford implemented these ideas in his plant and real life conditions influenced him to modify and improve his assembly methods.

Today, more refined methods of assembly have emerged. As a logical extension of basic assembly line principle, methods of replacing operators by mechanical means of assembly have been devised. These methods have rapidly gained popularity for mass-production and are usually referred to as automatic assembly. Automatic assembly serves the following advantages:

(i) Reduction in the cost of assembly.

(ii) Increased productivity.

(iii) Increased product consistency.

(iv) Decreased exposure of operators to hazardous operations.

An automatic assembly machine usually consists of a transfer system for moving the assemblies from workstation to workstation, automatic workheads to perform the simple operations, and inspection stations to check whether the various operations have been completed successfully.

However, the recent developments in robot technology will change the character of assembly lines in the future, when the problems with vision recognition of comoponent part orientation or in the development of economical, general purpose, part feeder-orientator systems are solved. Robotic assembly lines will begin doing the simple, repetitive assembly operations that are now done by workers, or automatic workheads.

Robots are machines that can duplicate human manipulative skills and flexibility with accuracy and

precision. They can be reprogrammed to do other operations, where forms of current automation are limited to a specific application. Robots can be thought of as off-the-shelf automation. They are controlled by computer software. Thus, the operation done by the robot can be changed almost instantaneously by simply changing the program controlling the robot.

The environment in which the assembly line with robots will operate will be vastly different from today's most sophisticated manufacturing facility. The environment will include automated storage retrieval systems in addition to computer generated shop schedules.

1.2. Problems in Assembly Process Planning

So far, only the historical development of assembly lines have been discussed, but also the planning/designing of an assembly line should be discussed.

There are five major factors of cosideration in assembly line design. They are:

1. Work bench design,
2. Material supply system,
3. Human element,
4. Work design,
5. Line balancing.

Due to the nature of the work and/or balancing consideration, some of the assembly tasks might not be prefered to be performed on the assembly line. Such tasks are

assigned to work benches and subassemblies produced at these work benches are fed to the assembly line. The allocation of work to and the design of work at those work benches constitutes a major production problem.

Second factor is related with material requirements planning and material handling.

Third factor aims at increasing the production efficiency by exploiting the psychological aspects of design. For examle, line speed, working conditions etc. all influence the productivity of the workers due to their impact on the psycholgy of the workers.

Fourth factor deals with the ways of generating productive work conditions based on ergonometry studies.

The last factor is perhaps the most essential component of the planning action, since it deals with the assignment of the minimum rational work elements on the line to the workers in such a way that the product is manufactured in a required period with less cost. And it is this last factor that will be treated in the remainder of this thesis.

1.3. Importance of Assembly Line Balancing (ALB) in Industry

Since the times of Henry Ford, assembly processes have progressed very rapidly. Up to late sixties, the assembly line was prevalent for car and truck production, whereas today, assembly lines are extensively used for assembling consumer durable items, such as radios, TV sets and refrigerators. Interchangable parts are assembled at a

set of sequential work stations where at each workstation a prespecified part of the total work content is performed. The assembly is usually moved by mechanical means, e.g. by a belt, a conveyor or an indexing line. Provided that parts are available when they are needed, and that demand for the product is adequate, these traditional assembly lines can be highly efficient. Balancing aims at efficient assembly lines. The well-balanced sequence of operations reduces manual handling and work-in-process inventories, since logic behind the balancing does not permit subassemblies to be produced more than the final assembly line can handle, and the operation that needs some kind of materials are located near to each other on the line.

ALB is an important production planning function, because the efficiency of an assembly line is directly linked to the quality of the balance. It is a job requiring knowledge of the product, layout, process, materials, tools, labour, and rules for combining this information. In addition, every company should use its own fast accurate method representing its own conditions.

Whatever methods are employed, there are a multitude of companies involved in assembly products and there is a real need for balancing techniques that reduce time of preparing balances and improve the quality of balanced line itself. The importance of this need becomes even more emphasized when one considers the dynamic nature of the industrial environment. The dynamic behaviour of the market

and the changes in technology force companies to react spontaneously to new situations. When demand flactuates, an immediate revision of production planning might be required. Especially, for mass-production, it is a challenging issue from the production planning and control point of view. Because any change of the production rate affects the production rate of all the parts and components going into the assembly, and thus leads to an imbalance of the whole system. That means the system operates with uncontrolled balance delays. Flactuations in demand also lead to increases in cost, if the issue of getting acceptable results are not taken into consideration very soon.

Because of these reasons, ALB became a main subject of intensive research in the past. And it is still sudied by many researchers today.

1.4. Definition of the ALB Problem

ALB problem is characterized by several challenging properties including discreteness and technological precedence among the operations. The objective is to minimize a function measuring idle time which is defined over a permutation group of work operations. The members of which are subject to technologically determined precedence relations on their possible temporal sequences. A feasible solution must satisfy a system of restricting inequalities on the sums of operation time values for combination of work elements.

While defining ALB problem, it is assumed that:

(i) The demand is certainly known.

(ii) Operations are deterministic. Stochasticity is not allowed.

(iii) Line is serial, no paralleling of tasks is allowed.

(iv) There is one and only one model being assembled on the line at a time.

The product to be assembled is comprised of many different parts. Because of the technical restricions, namely precedence relations among the operations, it is necessary to assemble these parts in some specific sequence or set of sequences; that is there is an ordering upon he sequence in time in which the parts may be assembled.

If the quantity of production for a particular period is specified, and if the production rate is to be uniform over that period, then the unit rate of production is determined. That is, the amount of time elapsing between successive units as they move along the conveyor is constant which is called the cycle time of the assemly line. Each assembly operator must be assigned a combination of tasks such that the sum of the times required to carry out these tasks is equal to or less than the cycle time. If his assigned work requires an amount of time greater than the cycle time, then he will not be able to perform all of his tasks or will be unable to maintain his position on the line and fall behind. Of course, if the duration of work assigned

to any operatoris less than the cycle time, he will be idle part of the cycle [2].

After ALB is described by Salveson as above, many researchers studied ALB problem intensively. Today, ALB is one of the most worked-upon problems in OR field.

The ALB problem can be described as:

Assigning a set of tasks to a set of workstations so that all precedence relations and cycle time constraints are satisfied and the objective function representing the sum of idle time at each work station is minimized.

The total idle time (TIT) can be determined as,

$$TIT = \sum_{j=1}^{m} (T_c - p_j) = mT_c - \sum_{j=1}^{m} p_j = mT_c - T_w \qquad (1)$$

where;

T_c : Cycle time.

T_w : Total work content.

P_j : Total work content of station j.

m : number of workstations.

Then the objective can be expressed as follows:

$$\text{Min } [\, mT_c - T_w] \qquad (2)$$

But since T_w is a constant, one can write it as,

$$\text{Min } [\, mT_c \,] \qquad (3)$$

So the objective function can be treated in three different ways:

(i) Minimizing the cycle time for a fixed number of

workstations. That is the case, when there is a fixed number of machine locations on the assembly line, and when operation times determine cycle time.

(ii) Minimizing the number of workstations for a fixed production rate.

(iii) Minimizing the product of the number of workstations and cycle time.

1.5. Computational Complexity of ALB Problem

ALB is a combinatorial problem. Because of its size and resulting computer storage requirements, ALB is unlikely to be solved using general integer programming methods. It has extremely large number of feasible solutions. For an n-task problem, if there are r precedence requirements, then there are roughly $n!/2^r$ distinct feasible sequences [3]. This combinatorial choice problem, therefore, posseses severe computational difficulties. So, most of the studies have concentrated on developing heuristic solution methods to solve problems of size met in practice.

1.6. Plan of the Thesis

In this thesis, a branch and bound (B&B) method for ALB is presented. In addition to the regular precedence and cycle time restrictions, several additional restrictions are added to the problem definition in order to make it more realistic. An upper bound (UB) concept is introduced which improves the computational efficiency of the B&B method

proposed. A computer program package is proposed and presented in detail.

This thesis consists of five chapters. Following introductory chapter, a literature survey on ALB is presented in chapter 2. In that chapter, Single Model Assembly Line Balancing (SMALB) is taken as the base problem, and others are considered as its extensions. In chapter 3, the proposed solution method is presented. The method is dicussed in detail. Several different versions of the method are developed and computerized. All of them are presented by an example. Computational results of the methods are also presented. Chapter 4 contains the implementation of the method on the mixed-model lines. Chapter 5 concludes the study.

II. ASSEMBLY LINE BALANCING PROBLEMS AND RELATED LITERATURE

2.1. Single Model Assembly Line Balancing (SMALB).

2.1.1. Mathematical Formulation

The assembly line balancing problem considered here is to minimize the number of stations along an assembly line to perform a given set of operations without exceeding the cycle time for the station and satisfying the precedence relations between the operations.

SMALB, as defined above, was formulated as a 0-1 integer programming model [4]. The mathematical program is as follows. But first, let us define the notation:

I : the task set , $I = \{1,2,\ldots,n\}$,

J : the station set, $J = \{1,2,\ldots,m\}$, where $m \leq n$,

t_i : The process time of task i , $i \in I$,

c_j : Penalty associated with using workstation j, $j \in J$.

P(i) : { Immediate predecessors of task i },

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

(i) Objective Function

$$Z = \sum_{i \in I} \sum_{j \in J} c_j x_{ij} \tag{4}$$

The objective function Z as defined above represents the cost of using the stations, and thus is to be minimized. Since c_j is the penalty associated with using workstation j and the objective is to use a minimum number of workstations, cost of using later stations should be significantly larger, such that;

$$c_{j+1} \geqslant M c_j \text{ , for every } j \in J-\{n\} \tag{5}$$

where M is a sufficiently large positive integer.

So the purpose of above objective function is to make later stations exceedingly costly and to assign the operations to the earliest station possible on the assembly line.

(ii) Constraints

a. Assignment Constraints

$$\sum_{j \in J} x_{ij} = 1 \text{ , } i \in I \tag{6}$$

These constraints guarantee that every task is assigned to one and only one station.

b. Cycle Time Constraints

$$\sum_{i \in I} t_i x_{ij} \leqslant T_c \text{ , } j \in J \tag{7}$$

These constraints imply that the total process time for all the tasks assigned to a station can at most be equal to the prespecified cycle time.

c. Precedence Constraints

$$x_{ik} \leqslant \sum_{j=1}^{k} x_{hj} \quad , \; i \in I \; , \; j \in J \; , \; h \in P(i) \tag{8}$$

These constraints guarantee that no task should be assigned before all of its predecessors are assigned.

d. 0-1 Constraints

$$x_{ij} = 0 \text{ or } 1 \quad , \; i \in I \; , \; j \in J \tag{9}$$

These constraints imply that no task can be split among two or more stations and thus each of the variables x_{ij} can take on values 0 or 1 only.

So the following 0-1 program represents a formulation for the single model assembly line balancing problem.

$$(4) \quad \text{Min } Z = \sum_{i \in I} \sum_{j \in J} c_j x_{ij}$$

$$(6) \quad \sum_{j \in J} x_{ij} = 1 \quad , \; i \in I$$

$$(7) \quad \sum_{i \in I} t_i x_{ij} \leqslant T_c \quad , \; j \in J$$

$$(8) \quad x_{ik} \leqslant \sum_{j=1}^{k} x_{hj} \quad , \; i \in I \; , \; j \in J \; , \; h \in P(i)$$

$$(9) \quad x_{ij} = 0 \text{ or } 1 \quad , \; i \in I \; , \; j \in J$$

In the above formulation, if m^* is the optimal number of stations needed, then the following bounds are applicable on it:

$$\sum_{i \in I} t_{i/Tc} \leqslant m^* \leqslant n \tag{10}$$

The bounds on m^* are referred to as the theoretical minimum and maximum number of stations needed and are denoted by m_{min} and m_{max} respectively.

Patterson and Albracht [5] developed some solution methods to the above 0-1 integer programming problem by making some simplifications. Based on the precedence relations, they define for each task i the earliest and latest stations that task i can be assigned to. This manipulation, together with the fact that one of the tasks with no followers in the precedence diagram must be assigned to the last station of the line, resulted in a new formulation which significantly reduced the size of the above formulation. Furthermore, in the objective function only the stations m_{min}, m_{min+1}, ..., m_{max} need to be considered, since, by definition, at least mmin stations are required.

Thangavelu and Shetty [6] improved the above formulation. Their mathematical model is as follows:

(i) Objective Function

$$\text{Min} \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij} \qquad (11)$$

where;

$$C_{ij} = \begin{cases} t_j \left[\sum_{h \in F} t_h + 1\right]^{(i-M_0-1)} & , \; i=M_0+1,\ldots,M \\ 0 \quad , \quad \text{otherwise.} \end{cases}$$

$$F = \{ j \mid P(j) = \emptyset \}$$

$$M_0 = \left[\sum_{i \in I} t_i / T_c\right]^+ \qquad (12)$$

with $\lceil a \rceil^{+}$ denoting the smallest integer greater than or equal to a.

This objective function makes the use of more than M_o stations very costly. Since later stations are very costly, this function forces to assign the operations to the earliest station possible. Since, at least, the first M_o stations must be used, they need to be assigned a cost.

(ii) Precedence Constraints

$$x_{ik} \leq \sum_{j=1}^{k} x_{hj} \quad , i \in I \ , j \in J \ , h \in P(i) \qquad (13)$$

Thangevelu and Shetty [6] showed that this inequality can be replaced by the following set of constraints thus reducing the number of constraints:

$$\sum_{j \in J} (m-j+1)(x_{ij}-x_{hj}) \geq 0 \quad , i \in I \ , j \in J \ , h \in P(i) \qquad (14)$$

They solved the problem using Balas's additive algorithm with some modifications to account for the special structure of the problem.

Cycle time, assignment and 0-1 constraints are the same as the expressions (7), (6), and (9) respectively.

2.1.2. Heuristic Methods for SMALB

Assembly line balancing is a combinatorial problem. The difficulty to find an exact solution for it comes from the fact that it can have extremely large number of feasible

solutions. Hence, construction of heuristics is of interest.

Heuristic methods seek to arrive at good solutions relatively quickly by the application of a great variety of priority rules for assigning tasks to the station which have some built-in logic. Researchers developing heuristic procedures intend;

(i) to make the ALB problem applicable in the industry by developing procedures easy to understand and apply,

(ii) to get a solution with less computational effort. Kilbridge and Wester [7] devised a simple manual method based on the manipulation of a table of the operations arranged according to the column order of the precedence diagram. Although guidelines are given, the method depends on the practitioner's skill and is therefore confined to small problems.

A class of the heuristic methods are based on filling of workstations by assigning the operations sequentially from a ranked list. These methods are usually referred to as serial methods and they differ only in the criterion they use for obtaining the ranked list of operatons. All the methods prepare lists with respect to some heuristic criteria. There are four basic ranking criteria:

(i) RPW (Relative Positional Weight) : The positional weight of a work element is its own processing time added to those of all following work elements.

(ii) TF (Total Number of Tasks Following) : For a particular work element, the TF value is the number of all

work elements following it.

(iii) LCR (Largest Candidate Rule) : For a particular work element, this criterion is its own processing time.

(iv) IF (Total Number of Immediate Followers) : For a particular work element, the IF value is the number of immediate followers.

The methods rank the operations according to one of the above criteria. The ranked lists are obtained by ordering the tasks in decreasing order of the criterion value. Ties are broken arbitrarily. The assignments are made starting from the top of the ranked list without violating precedence relations and any other requirement. Logic behind the ranking of the operations is to assign the most vital operations from the balancing point of view as soon as possible.

Helgeson and Birnie also devised 10-SP method [8] which simply selects the best of ten solutions, each obtained by using a different ranking system for the selection of work elements. Ten ranking systems were obtained from the combination of the four basic ranking methods. The 10-SP algorithm solves each balancing problem 10 times, using in turn, each of the above ranking methods. The solution yielding the least balance delay becomes the 10-SP solution.

Another class of heuristic methods have resulted from the attempts of making some exact solution procedures computationally feasible by the introduction of heuristic rules leading to a suboptimal solution.

Mansoor [9] improved his "backtracking" method since it could lead to very lengthy calculations on larger problems, although it guarantees an optimal solution. The method called MALB is based on the optimum-seeking iterative method. The method minimizes the cycle time for a given number of stations. It aims at extending the feasible sequence until all the operations in the assembly are assigned and grouped into the required number of stations. If the feasible sequence can not be extended, the algorithm applies a backtracking procedure, where the last assignment is cancelled and a systematic attempt is made to extend the sequence until either all the operations are assigned or it is decided that no feasible solution exists. In the latter case, the cycle time is increased by one unit and the assignment procedure continues to get an optimal solution.

The computation time limits a wider application of the optimum-seeking algorithm. Although the algorithm may be used in solving large assembly problems with low F-ratios which is the ratio of number of ordering relations to the possible number of ordering relations, its general application is restricted and it was increasingly evident that the answer lay in the development of heuristic models that could effectively limit the backtracking iterations. For such a model, Mansoor [9] introduced four heuristic rules that limit,

(i) the slack available at each iteration,

(ii) the backtracking within a station,

(iii) the backtracking between stations,

(iv) the total backtracking in a problem.

MUST, MUltiple Solution Technique of Mansoor and Rubinovitch [10] is based on the optimum seeking method of Mansoor-Yadin algorithm. The procedure generates alternative solutions of equal quality for SMALB. Mansoor-Yadin algorithm minimizes the cycle time for a given number of stations. It begins with the minimum theoretical cycle time. The balancing procedure consists of a sequential addition of work elements to admissible subsets. It begins with the empty set, then the work elements are added one at a time from the appropriate set of immediate followers (i.e. belonging to the subset being extended) until all subsets are complete with respect to the first station (i.e. they are feasible first station assignments); Each complete first station assignment is then extended ina similar manner to form complete second, third, etc. station assignments. The optimum-seeking algorithm exhaustively executes the extension and recording process discussed earlier and eventually, the entire set of optimal solutions is generated. Because of memory and time limitations, this exact solution algorithm have been modified by Monsoor and Rubinovitch through heuristic rules which limit the search for larger problems, to give a heuristic procedure for generating multiple solutions. Three heuristic rules are used for that purpose:

(i) The size of circular buffer and handling overflow.

(ii) The total number of extensions from each station.

(iii) The total number of extensions from each assignment.

The method proposed by Baybars [3], namely LBHA-1, has five phases. In phase 1, the size of the problem is reduced, whenever possible, by eliminating certain tasks. Phase 2 identifies mutually exclusive assignment decision;s. In phase 3, the problem is decomposed into smaller problems whenever possible. In Phase 4, any collections of tasks likely to be assigned to the same station are identified, and, finally in Phase 5, a solution is sought for the problem using various heuristics. The assignment procedure in this phase is a backward procedure, starting with the last tasks in the precedence diagram and basing the assignment decisions on the principle that " last tasks are likely to be assigned to the last stations along the line ". Several tie-breaking rules are used in the procedure. If there are two candidate tasks to be assigned, the tie is broken by using the largest operation time rule. Moreover, if a tie cannot be broken this way, it is broken by choosing the task with the largest number of unassigned predecessors.

Pinto, Dannenbring and Khumawala [11] developed a method called Heuristic Network Procedure (HNP). The procedure uses simple heuristic rules to generate a network which is traversed using a shortest route algorithm. The procedure first generates all possible nodes corresponding to feasible station sequences, which are found by using simple heuristic methods such as RPW, LCR, or random assignment. Each heuristic can be used to generate a solution which is

equivalent to a set of nodes in the directed network. After each heuristic is applied independently, the sets of nodes can be combined to form a composite network. This allows consideration of additional arcs and as a result the possibility of a shorter path and, hence, a better solution.

2.1.3. Optimal Solution Techniques for SMALB

The earliest attempts at optimizing line balancing relied upon rigorous mathematical models. But most of these have not yielded a really practical method even when using a computer owing to the exhaustive nature of calculations.

The most efficient 0-1 integer programming solution method for SMALB is presented by Thangavelu and Shetty [6]. The method is mainly based on Geoffrion's version of Balas' Method with some simplifications or eliminations to adopt the special structure of SMALB problem.

Held, Karp and Sharehsian [12] developed a dynamic programming method to optimize the number of stations for a given cycle time. The algorithm calculates the cost of feasible sequences and the sequence with the minimum cost is the sequence that minimizes the number of stations. A feasible sequence is a subset of tasks that can be executed in the indicated order without any other tasks being done. The cost of a feasible subsequence is the number of filled-up stations it requires plus the time in the last station. The method finds the sequence with the lowest cost, which is the optimal solution to the problem.

Jackson's method [12] is an exception to the other modelling-type solution methods in that it requires extensive computation. It is excellent for hand calculations for upto thirty tasks. Above that it can be programmed, although it requires large amount of computer time. The idea behind it is simple: Construct all feasible first workstations;then for each such first wokstation, construct all feasible second workstations; for each first-second combination, construct all feasible third workstations and so forth. At some point, after k stations are constructed, it will be found that one or more of the assignments cover all the tasks; therefore the method minimizes the number of workstations for a given cycle time. The method makes use of a dominance relationship between feasible sequences. A feasible sequence is dominated by another feasible sequence if the two sequences can be assigned to stations so that both sequences have the same tasks performed at the same station. Thus, both feasible sequences result in the same assignment.

Assche and Herroelen [13] developed an optimal solution method similar to Jackson's method. The procedure is a B&B algorithm. The method is equipped with dominance rules, bounding arguments and reliable heuristics. Assche and Herroelen also introduced a dominance rule to the B&B procedure by which multiple solutions and inferior solutions are eliminated from further consideration. The method is a tree search procedure. Each iteration of this procedure begins with a node representing the assignment of work

elements to a single workstation. Once a node is formed the remainig work elements are checked whether they all can form a station, namely the last station, to yield an immediate solution. This is a heuristic to get a feasible solution as fast as possible. If it is determined that no immediate solution can be found, the procedure branches into a number of descendant nodes corresponding to the feasible undominated next station assignments and computes for each such node a lower bound. The proedure then chooses the node with the smallest lower bound for the next iteration.

A more complete survey of exact methods for SMALB problems can be found in Baybars [4].

2.2. Mixed-Model Assembly Line Balancing (MIXALB)

Multimodel lines are lines on which two or more models of a product are assembled separately in batches, whereas mixed-model lines are ones on which two or more models of a product are assembled simultaneously.

An obvious approach to line balancing in both cases is to balance each model seperately. It requires successive applications of single model assembly line balancing. This is valid for multimodel lines, since there will be one and only one model on the line at a time, whereas mixed-model lines lead to some difficulties when applying this procedure. With this method, the solution yields equal number of stations alloted for each model and excessive amounts of idle time in the last stations.

A more serious difficulty is that a given task may be assigned to different stations for different models. That increases set-up and material handling costs. In order to ensure that, for a given model mix, every repetition of a given task will be assigned to only one station, compound precedence and aggregated duration concepts, which are obtained by the superposition of precedence diagrams and operation times respectively, are introduced.

There is another difficulty in any cycle time variations as the work load depends on different models being in-phase. So another criterion is to equalize the workload on operators over a convenient time interval, e.g. over a shift. The line is thus balanced for a single model but on the basis of a convenient time interval requirement.

Hence, mixed-model line balancing problem has two main aspects :

(i) The assignments of tasks to operators in order to minimize the number of stations and consequently labour cost by balancing the work content among the stations over a time period, such as a shift [14].

(ii) Sequencing of models to improve assembly performance and decrease total set-up cost [15].

Thomopoulos [14] showed that solutions to first part can be obtained by any single model assembly line balancing algorithm by replacing cycle time with shift time, task times with aggregate task times, and precedence with compound precedence. Unfortunately, there is a high possibility that

the solution may not be smooth if the models are considered separately. That is, aggregate task times on the compound precedence diagram may cause workloads of stations vary significantly for different models. That is why, Thomopoulos suggested exchange of operations between the stations, after getting an initial feasible solution, by considering for each model and station how close to station time for a model is to the mean station time for that model.

2.3. Assembly Line Balancing With Paralleling

One of the main assumptions in the ALB problem is that the line is "serial" with no paralleling of tasks allowed, i.e. each task is assigned to a particular workstation and no two workstations can perform the same task. The serial line assumption restricts the cycle to be at least equal to the maximum task time, and hence limits the production rate. There are several alternative ways of achieving a higher production rate, eg. the use of over time, another assembly line, subcontracting, buffer stocks, and paralleling. The paralleled line may, in many case, be a less costly method of increasing production than the other alternatives.

Paralleling can be of two types [16] :

(i) Paralleling of tasks.

(ii) Paralleling of stations.

In both cases, the main problem is selecting the tasks to be paralleled, in such a way that the total relevant

costs are minimized while satisfying the production requirement [17].

Paralleling provides the following advantages [18]:

(i) The primary reason for paralleling is to increase balance efficiency. When a station is added parallel stations then the cycle time for the set of parallel stations is obtained by multiplying the cycle time by the number of parallel stations in that set. Thus, the likelihood of a good fit increases. This is particularly relevant when balance difficulty is encountered due to larger work elements being of the same order as the cycle time.

(ii) Parallel stations enable the production rate to be greater than the limitation imposed by the longest work element. Output can then be increased relatively smoothly to meet demand.

(iii) Incorporating parallel stations in a line may lead to a substantial reduction in idle time incurred due to differences in operator process times, the human factor.

(iv) The variability of the operation times are reduced as a result of ensuring a regular workflow without excessive layout difficulties and transportation times.

Pinto, Dannenbring and Khumawala [16] developed a method applicable to both types of paralleling problem. Both problems are formulated as a mixed integer programme and a branch and bound method is applied for its solution. The method proceeds by partitioning the sets of all combinations into subsets of partial combinations.

A full combination is achieved by fixing all tasks either as to be paralleled or not. The case where none of the tasks are to be paralleled is solved first to obtain an initial solution. If this solution is feasible, i.e. it meets the required production rate, then it constitutes an upper bound on the total cost. If this is feasible, it is the optimal solution [17].

Buxey [18] developed a computer program that handles parallel stations. The approach is based on a sophisticated version of RPW method and random generation method.

2.4. Assembly Line Balancing With Stochastic Operation Times

In standard single model assembly line balancing, task times are assumed to be deterministic. In practice, the assumption that task times are known constants is at best a simplifying approximation. There are cases in which the variabilities of task times are rather large relative to their means [19]. This is especially true when tasks are complex and require high level of skill and concentration. Under these circumstances, the deterministic assumption is inadequate and solution procedures are not of much help. When task times are stochastic, the issues involved are more complex and objectives varied. For the case when task times are assumed to be independent normal variates, several authors have introduced different optimality criteria and proposed various heuristic line balancing procedures [19].

When the task times, t_i , are independent random variables with means, m_i , two formulations have been proposed [20] :

F1. Min (n) (15)

s.t.

$$M_j \leq bT_c \ , \ j \in J \tag{16}$$

where;

M_j : Average work content at station j,

b : Maximum proportion of cycle time to be utilized.

F2. Min (n) (17)

s.t.

$$P \{ p_j \leq T_c \} \geq q \ , \ j \in J \tag{18}$$

where;

q : Minimum probability for the station time not exceeding the cycle time.

Sphicas and Silverman [20] showed that for certain classes of task completion time distributions, the deterministic and stochastic formulations of the assembly line balancing can be transformed so that they are equivalent to each other. This is significant in that the heuristic procedure of assigning tasks to only a predesignated proportion of the cycle time has theoretical justification, and the corresponding probability that all tasks will be completed within the cycle time can be induced from this transformation. In addition, for these classes of

distributions, optimum balances can be found for stochastic problems using less complex and more efficient deterministic algorithms.

Kao's approach [19] is for the solution of second type of formulations. He used a dynamic programing approach similar to that proposed by Held and Karp dealing with the case in which task times are assumed to be deterministic. Under stochastic formulation, the optimal return function is a vector and the recursive method for finding it, is based on Mitten's preference ordering dynamic programming. Like most dynamic programing algorithms, the procedure is useful only for problems of limited size due to the fact that storage and computation requirements grow very rapidly as the number of tasks in an assembly line increases [18].

2.5. Commercial Computer Packages for ALB

The dynamic behaviour of the market forces the companies to react quickly to new situations. They require to change the production scheduling and working instructions due to these changes. In order to overcome this difficulty many sophisticated computer programs have been developed.

Computer-aided ALB offers many advantages in that it enables for more possibilities to be investigated and thus increases the chance that good solutions will be found. It also enables the planner to obtain balances for differing production rates and to prepare contingency balances for unexpected decreases in the number of workers due to sickness

or absenteeism. In addition, with the widespread availibility of time sharing facilities, it is now possible to put much of planners' experience for good use by enabling him to interact with the line balancing program via a small computer terminal. For these purposes, several computer programs have been developed.

Montgomery et al. [21] mentioned some of the pioneers of computer programs for ALB as follows:

COMSOAL (Computer Method of Sequencing Operations For Assembly Lines) is a method developed at Chrysler Corp. and reported by Arcus in 1966. The procedure is to iterate through a sequence of randomly generated alternative solution and to keep the best one. CALB(Computer-Aided Line Balancing) is introduced by Advanced Manufacturing Methods Program(AMMP) of IIT Research Institute in 1968. The program starts by sorting the tasks according to their task times and precedence requirements. Based on this sort, operations are assigned to stations so that the solutions obtained by CALB are described as near-optimum and are achieved with only a few seconds of time on the computer. ALPACA (Assembly Line Planning and Control Activity) is developed by GM in 1967. ALPACA is an interactive ALB program having a batch input system. The system allows the users to transfer work from one station to another along the flow line and immediately assess the relative efficiency of the change.

Some of the more well-known computer programs being used will be briefly introduced below.

NULISP - Nottingham University Line Sequencing. Program

NULISP [22] is designed similar to a computer language for ALB problems. It has programs available in batch or interactive mode and their flexibility enables them to be used for the day to day sequencing of tasks on a flowline production system and to aid the designer when planning the positioning of facilities and setting of team sizes for future production lines. NULISP uses heuristics, a weighted random selection procedure, to generate a sufficient number of solutions that there is a high probabilty of obtaining an optimal result.

NULISP can handle the following:

(i) Minimizing either number of stations or cycle time,

(ii) Processing stochastic operation times,

(iii) Grouping, negative or positive, of tasks,

(iv) Fixing of tasks at particular stations,

(v) Processing parallel stations,

(vi) Processing either single model or mixed model problems.

The modified interactive version of NULISP has been called IDEAL - Interactive Design and Evolution of Assembly Lines. Using IDEAL system the user may judge the consequences of data changes, e.g. cycle time or restrictions; and reevaluate an assembly line within minutes.

INTAKT - Interactive Line Balancing

Bullinger et al. [23] developed an interactive method which is based on the experience gathered with a batch program, being applied in several companies for more than five years in Germany. The program allows user to make his own assignments by himself. Meanwhile,the program only helps him by a control mechanism that checks the consistency within the input data, such as line specifications, work contents etc. and the assignments being made.

III. AN OPTIMAL SOLUTION TECHNIQUE , B&B METHOD FOR ALB

3.1. Problem Definition

Conventional ALB is defined as the procedure of assigning a set of tasks to a set of workstations so that all precedence relations and cycle time constraints are satisfied while an objective function represented by the number of workstations, cycle time or their product is minimized. This definition is a rather simplified representation of the actual situation met in practice. It considers only the precedence and cycle time restrictions. So any solution procedure with a chance for implementation should start from a more realistic problem definition.

The new and improved B&B method to the ALB problem considers most of the special characteristics of assembly lines in real-life. Following restrictions completes the conventional definition of ALB for industrial purposes.

3.1.1. Technological and Managarial Restrictions Imposed on ALB

(i) Orientation Constraints

$$\sum_{l=1}^{i-1} (S_i x_{ij} - S_l x_{lj})\, S_i S_l x_{lj} = 0 \quad , \quad i \in A^k \quad , \quad j \in J \quad , \quad k=1,2 \qquad (19)$$

where;

A^k : set of operations having orientation k.

S_i : Orientation of task i, S_i = 0, 1 or 2

When large products are assembled physically distinct working areas may occur on the assembly line. Every operation is performed in one of those working areas. If two or more operations needing different working areas are assigned to a single worker, this will require positional change of the worker thus resulting in non-productive labour. This is undesired from the line balancing point of view. For example, on a refrigerator assembly line, some operations can be performed facing only back side of the product and some facing only the front side of it. There are also some operations which may be performed from any orientation.

Meanwhile, on an automobile assembly line, number of working areas may be much more, such as front, back, left-hand side, right-hand side, etc. [24]. Another aspect is to determine the relations between those working areas. They should be defined clearly to point out the cases where positional change is not necessary between any pair of them.

(ii) Strict Grouping of Operations

$$x_{ij} + \left(\sum_{h \in U_1^k} x_{hj} \Big/ |U_1^k|\right) \leq 1 \quad , \; i \notin U_1^k, \forall j, \forall k \qquad (20)$$

where;

U_1^k : set of grouping 1 elements , $k=1,2,\ldots,K_1$

Technological or managerial considerations may force a group of operations to be assigned to the same or adjacent stations where no other operation not a member of the group is allowed to be assigned with them. These operations have to

be performed next to each other subject to their own precedence requirements. For example, some groups of operations have to be completed immediately once they are started. This may be the case when noisy or dirty operations are isolated from others.

(iii) Assignment to the Same Station

$$\sum_{i \in U_2^k} x_{ij} = \left|U_2^k\right| x_{hj} \quad , \quad h \in U_2^k \ , \forall j \ , \forall k \qquad (21)$$

where;

U_2^k : set of grouping 2 elements , k=1,2,..,K_2

With this restriction, a group of operations are required to be assigned to the same station, while additional operations may be assigned to that workstation if feasible to do so. For example, it might be required that placing and tightening of screws be performedin the same workstation since otherwise they may drop as the product move from one workstation to another.

(iv) Prefered Grouping of Operations

This is that similar to the first grouping restriction except that operations not a member of the group can be assigned together with the group elements to the same station, if feasible to do so. The aim is to complete the assignment of a group of operations within the least number of stations adjacent to each other. This may be the case, for example, when a group of operations requires similar

materials or equipments. This can not be represented as a constraint, because it is a preference rather than an absolute requirement.

(v) Strict Assignment to Different Stations

$$x_{ij} + x_{hj} \leq 1 \quad , h \in U_4^k - \{u^k_{41}\}, \forall j, \forall k, i = u^k_{41} \qquad (22)$$

where,

U_4^k : set of grouping 4 elements , $k=1,2,\ldots,K_4$

The assignment of two or more operations together to a single station might be undesirable due to technological or managerial reasons. Those operations should be assigned to different stations. For example, some operations might require intensive concentration so that the assignment of additional operations of similar kind to that same operator might cause fatique of that operator.

(vi) Complementary Operations

$$x_{ij} = x_{h\,(j+1)} \quad , i = u^k_{51} \; , h = u^k_{52} \; , \forall k \qquad (23)$$

where;

U_5^k : set of grouping 5 elements , $k=1,2,\ldots,K_5$

The operations are said to be complementary operations if they are performed by two operators simultaneously. In other words, with a given technology, the

performance of a task requires two operators and the task is split into two complementary operations to be performed simultaneously by two operators. In this case, considering synchronization, complementary operations have to be assigned to stations facing each other. That is, both operators should be able to perform those operations in same time interval.

So the following non-linear program represents the formulation for the extended single model assembly line balancing problem.

$$(4) \quad \text{Min } Z = \sum_{i \in I} \sum_{j \in J} c_j x_{ij}$$

$$(6) \quad \sum_{j \in J} x_{ij} = 1 \quad , \; i \in I$$

$$(7) \quad \sum_{i \in I} t_i x_{ij} \leqslant T_c \quad , \; j \in J$$

$$(8) \quad x_{ik} \leqslant \sum_{j=1}^{k} x_{hj} \quad , \; i \in I \; , \; j \in J \; , \; h \in P(i)$$

$$(9) \quad x_{ij} = 0 \text{ or } 1 \quad , \; i \in I \; , \; j \in J$$

$$(19) \quad \sum_{l=1}^{i-1} (S_i x_{ij} - S_i x_{lj}) \, S_i S_l \, x_{lj} = 0 \quad , \; i \in A^k \; , \; j \in J \; , \; k=1,2$$

$$(20) \quad x_{ij} + \left(\sum_{h \in U_1^k} x_{hj} \Big/ \left| U_1^k \right| \right) \leqslant 1 \quad , \; i \notin U_1^k, \forall j, \forall k$$

$$(21) \quad \sum_{i \in U_2^k} x_{ij} = \left| U_2^k \right| \, x_{hj} \quad , \quad h \in U_2^k \; , \forall j \; , \forall k$$

$$(22) \quad x_{ij} + x_{hj} \leqslant 1 \quad , \; h \in U_4^k - \{u^k_{41}\}, \forall j, \forall k, i = u^k_{41}$$

$$(23) \quad x_{ij} = x_{h\,(j+1)} \quad , \; i = u^k_{51} \; , \; h = u^k_{52} \; , \forall k$$

3.2. Solution Method

The algorithm has two phases. In PHASE1, Heuristic Assembly Line Balancing Program (HALB) generates an upper bound for B&B Method. In PHASE2, Optimum Assembly Line Balancing Program (OPALB) gets the optimum solution by B&B Method. Both phases are discussed in detail in the following pages.

3.2.1. Phase 1 - HALB

HALB consists of three heuristic methods, namely;

(i) RPW - Relative Positional Weight Method

(ii) LCR - Largest Candidate Rule Method

(iii) TF - Total Number of Tasks Following.

It first prepares the ranked lista of the operations for each method and then makes assignments according to those lists. Precedence relations, cycle time constraint and technological restrictions. The main output of this phase is the minimum number of stations among three solutions. This solution is an input for Phase 2 - OPALB. It is used as an upper bound for B&B method for relevant level.

3.2.2. Phase 2 - OPALB

This is the main program that solves the ALB problem to get an optimal solution (or multiple optimal solutions). There are two versions of this phase; one gets multiple optimal solutions whereas the other gets a unique optimum

solution. The only difference comes from their different fathoming criteria.

In general, B&B method is a tree-structured method. Each node on the tree represents a solution for the relevant level of the tree. And there is a lower bound associated for each node. Improvement of the solution is realized by branching from the leave node that has the smallest lower bound.

The algorithm generates possible station assignments, namely nodes, at each level of B&B tree. Then the question arises: If this node is accepted as the final assignment for related station level, what will be the minimum number of stations at the end? The answer to this question is the lwer bound of that node. Lower bound is the number of stations obtained by relaxing all the constraints except the cycle time constraint. Thus, in general, lower bound solutions are infeasible for the whole problem. The procedure is based on the conventional B&B logic of searching for a feasible lower bound which is the optimal solution. That is why, at each leave node, HALB is run for unassigned operations to check the feasibility of lower bounds. HALB generates an upper bound for each leave node to obtain the equality of upper and lower bounds.

The algorithm also eliminates some nodes by making use of a dominance rule and multiple solution check.

3.2.3. Aggregate Algorithm

Step 1 . Set level = 1

Step 2 . Generating possible station assignments or current level. Calculate lower bounds (LB) and penalties (P).

Step 3 . Run HALB module or each leave node to generate an upper bound (UB) for related nodes.

Step 4 . If UB = LB then go to Step 9.

Step 5 . Fathoming Rule: fathom the inferior branches or multiple optimum solution check (i.e. this part is dependant on the version implemented) of the algorithm.

Step 6 . Branching from the most appropriate node.

Step 7 . Set level = branched level + 1

Step 8 . If all operations are not assigned then go to Step 2.

Step 9 . Save the result into a text file and STOP.

Generation of possible station assignments is the only assignment part of the algorithm. The program first scans the network for preparing eligible activity set, so called pend list. It is the set of all assignable operations whose predecessors have already been assigned in previous nodes in the branch that is being improved. During assignment process, when an operation is assigned, the pendlist is continiously updated. Its algorithm is as follows:

Step 1 . a. Construct pendlist of size, say K,
i.e. number of operations in the list.

b. Set pointer = 1, i.e. pointing to the first element of the pend list.

c. Set assignment set empty.

Step 2 . Pick the operation that pointer points.

Step 3 . Test for feasibility:

a. If cycle time constraint is violated then go to Step 6.

b. If the orientation of the operation is not consistent with the orientation of the station, then go to Step 6. (Note that the first assigned operation determines the orientation of the station.)

c. If one of the technological restrictions is violated then go to Step 6.

Step 4 . Assign the operation, add it into the assignment set.

Step 5 . Update pend list, i.e. add the successors of the assigned operation to the pend list, if all of its predecessors have been assigned.

Step 6 . a. Set pointer = pointer + 1

b. If pointer less than or equal to K then go to Step 2.

Step 7 . Generation of a new node. Assignments should be added to the tree as a node.

a. First of all, test whether this assignment set is a subset of any other node in the same station

level being assigned. If it is a subset go to Step 8.

b. Add the set as a node in the tree. Calculate lower bound and penalty for this node.

Step 8 . a. Remove the last assigned element in the asignment set.

b. Update pend list, i.e. remove the successors of the last removed element from the pend list.

c. Set pointer so that it points to the operation that is removed.

Step 9 . a. Set pointer = pointer + 1

b. If pointer less than or equal to K then go to Step 2.

Step 10 . If assignment set is not empty then go to Step 8, otherwise STOP.

Lower bound and penalty are calculated as indicated below using the following notation.

N_r : Least number of stations for unassigned operations.

N_p : Current station level, number of stations for assigned operations.

LB : Lower bound.

P : Penalty.

Then,

$$LB = N_r + N_p \tag{24}$$

$$P = (TWC-T-k) / N_p \tag{25}$$

where,

$$N_p = [(TWC-T) / Tc]^{+} \qquad (26)$$

and k is a constant which Assche et al. suggested or more effective penalty calculation. They also suggested k to be chosen from the interval [1, $\sqrt{c}$]. The algorithm takes k as unity.

Fathoming rule is the only different part for two versions of program. The rule for the version getting unique solution fathoms the branches that are inferior to others. That is, set of assignments of a branch is a subset of any other branch with same station level with larger or equal lower bound. In this process, smoothing factor which is the smooth workloads over stations is ignored. The main idea is to get an optimal solution as soon as possible.

Both rules also fathom the branch whose lower bound is greater than the upper bound.

Branching from the most appropriate node needs a typical B&B search. The algorithm branches from the node that has minimum lower bound. Some secondary branching rules are also used when tie occurs in choosing minimum lower bound. The algorithm is as follows:

Step 1 . Start searching from the root of the tree.

Step 2 . Advance to next level.

Step 3 . If the node is fathomed or it is not a leave node, then go to Step 2.

Step 4 . Choosing the branching node;

a. If minimum lower bound is obtained then specify the node as branching node and go to Step 5. If tie occurs then go to Step 4.b.

b. If minimum penalty is obtained then specify the node as branching node and go to Step 5. If tie occurs then go to Step 4.c.

c. If maximum number of operations assigned is obtained then specify the node as branching node and go to Step 5.

Step 5 . If all the nodes on the tree are not scanned then go to Step 2, otherwise STOP.

3.3. Example Problem

The example problem which is a 10-task problem is solved for $T_c = 11$. (see Figure 1)

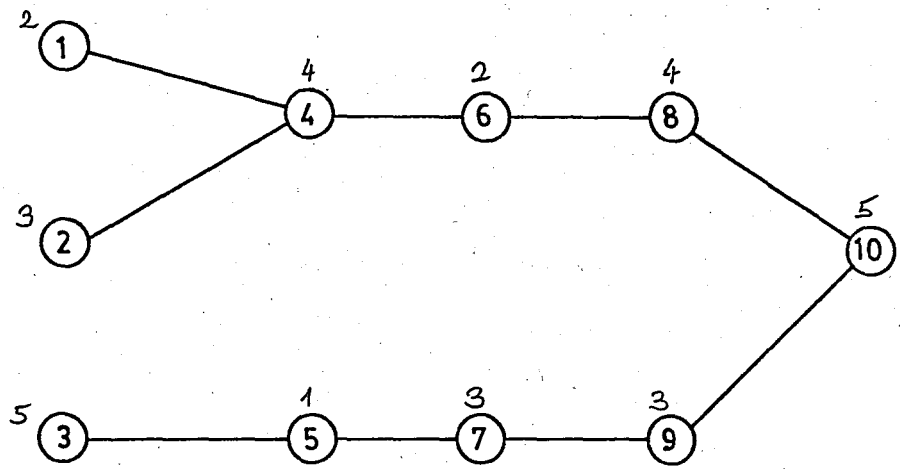

Figure 1. Precedence diagram of the example problem.

The algorithm first starts with generating possible station assignments for the first level, namely for the first station. Those possibilities which construct the B&B tree,

Figure 2, are obtained according to the algorithm given in section 3.2 and steps of the procedure for generating possible first station assignments are demonstrated in Table 1.

After obtaining the nodes for the relevant level, namely nodes 1, 2 and 3, the algorithm calculates lower bound and penalty for each of them.

$LB1 = [(32-11)/11]^{+} + 1 = 2 + 1 = 3$

$LB2 = [(32-11)/11]^{+} + 1 = 3$

$LB3 = [(32-9)/11]^{+} + 1 = 4$

$P1 = (32-11-1)/2 = 10$

$P2 = (32-11-1)/2 = 10$

$P3 = (32-9-1)/2 = 11$

By running HALB module following upper bounds are obtained:

$U1 = 4$, $U2 = 4$, $U3 = 4$.

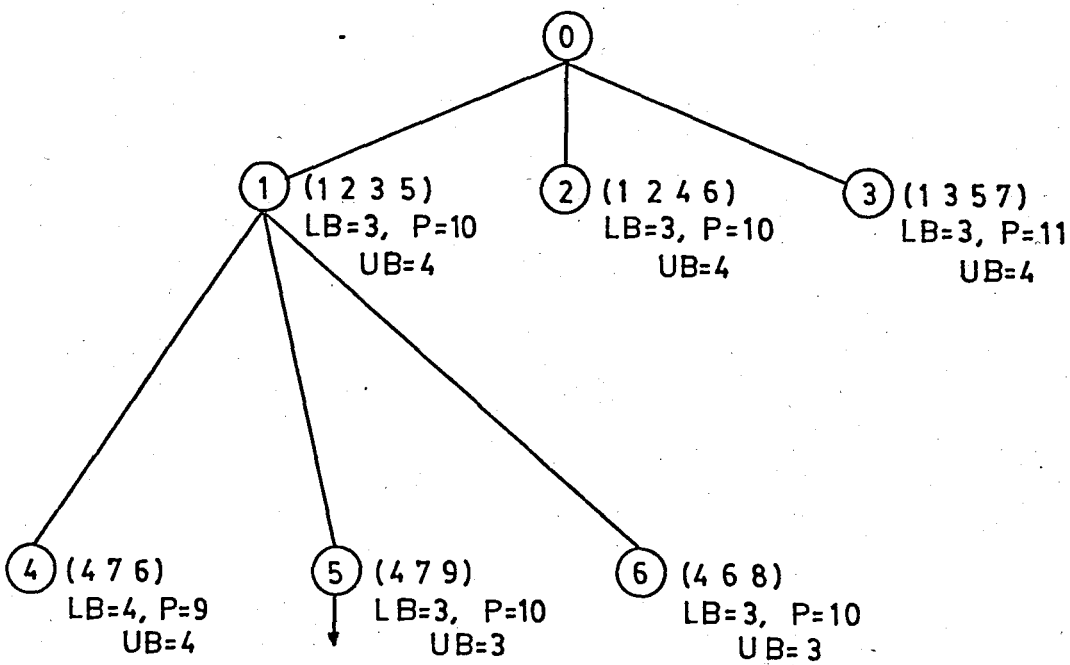

Figure 2. B&B tree for the example problem.

Since equality of lower bound and upper bound is not obtained for any node, that is none of the lower bounds could be shown to be feasible, the next step is to check whether any of the branches can be crossed off due to similarity with higher lower bound. Nodes 1, 2 and 3 have different sets of assignment. Hence, the algorithm branches from node 1 according to the branching rules given in section 3.2.2.

Nodes 4, 5 and 6 are generated in the same manner and lower bounds, penalties and upper bounds are attached in Figure 2. Lower bound is equal to the upper bound for node 5. So the program does not continue to branch further. It finds the final solution by tracing back on the tree, i.e.

Station 2 = { 4, 7, 9 }

Station 1 = { 1, 2, 3, 5 }

and by ading the HALB solution for unassigned elements to this partial solution, i.e.

Station 3 = { 6, 8, 10 }.

Table 1. Generation of possible first station assignments.

Pendlist	Assignment set	Time	Nodes
1,2,3	1	2	
1,2,3	1,2	5	
1,2,3,4	1,2,3	10	
1,2,3,4,5	1,2,3,5	11	{1,2,3,5}
1,2,3,4,5	1,2,3	(subset)	
1,2,3,4	1,2,4	9	
1,2,3,4,6	1,2,4,6	11	{1,2,4,6}
1,2,3,4,6	1,2,4	(subset)	
1,2,3,4	1,2	(subset)	
1,2,3	1,3	5	
1,2,3,5	1,3,5	6	
1,2,3,5,7	1,3,5,7	9	{1,3,5,7}
1,2,3,5,7	1,3,5	(subset)	
1,2,3,5	1,3	(subset)	
1,2,3	1	(subset)	
1,2,3	2	3	
1,2,3	2,3	8	
1,2,3,5	2,3,5	(subset)	
1,2,3,5	2,3	(subset)	
1,2,3	3	5	
1,2,3,5	3,5	6	
1,2,3,5,7	3,5,7	(subset)	
1,2,3,5,7	3,5	(subset)	
1,2,3,5	3	(subset)	
1,2,3	$\emptyset$		

3.4. Computer program

Computer program is developed in modular form for microcomputer usage. It is developed for IBM/PC and compatible machines under MS/DOS version 2.11. The language used is Turbo Pascel version 3.0. The program also implemented on a mainframe ,namely CDC Cyber 815.

The program limits are as follows:

(i) Maximum 100 operations on the assembly line.

(ii) Maximum 20 successors for each operation.

(iii) Maximum 10 blocks for each special grouping.

(iv) Maximum 15 operations for each block of those.

The computer program is developed in modular form and it consists of four modules.

1. INPUT - Data Manipulation Module
2. HALB - Heuristic ALB Solution Module
3. OPALB - Optimum ALB Solution Module
4. REPORT - Report Module

3.4.1. INPUT - Data Manipulation Module

This module prepares the data for the line balancing. Program occupies a capacity of roughly 46 KB's in external storage with about 1500 lines of source element. It has the following properties:

(i) Providing interactive data input/update,

(ii) Getting data from keyboard or file,

(iii) Saving each data in different files,

(iv) Conversion of network. Main programs - HALB and OPALB - work on a network whose arcs are directed from smaller node numbers to larger ones. Program provides the renumbering of the nodes appropriately for program usage. But inputs and outputs will be reported back according to user's numbering system.

(v) Providing data consistency check. At any time, user can check for;

a. Consistency of operation times with cycle time, i.e. paralleling is not allowed.

b. Consistency of times of U1 elements with cycle time, i.e. their sum can not be greater than cycle time by definition.

c. Consistency among the elements of all grouping sets, so that;

$$U_1 \cap U_2 = \emptyset$$

$$U_1 \cap U_3 = \emptyset$$

$$U_1 \cap U_4 = \emptyset$$

$$U_1 \cap U_5 = \emptyset$$

$$U_2 \cap U_4 = \emptyset$$

where;

U1 : set of grouping 1 elements,

U2 : set of grouping 2 elements,

U3 : set of grouping 3 elements,

U4 : set of grouping 4 elements,

U5 : set of grouping 5 elements.

d. Consistency within the orientations the elements of each grouping, namely U1, U2 and U3, i.e. the orientations should be the same for the orientations within any grouping.

e. Consistency within the orientations of U5 elements.

f. Connectedness of the precedence network.

g. Closed loops in the network.

3.4.2. HALB (Heuristic Assembly Line Balancing)

The program uses the algorithm presented in 3.2.1. Program occupies a capacity of roughly 22 KB's in external storage with about 750 lines of source element.

3.4.3. OPALB (Optimum Assembly Line Balancing)

The program uses the algorithm presented in 3.2.2. Program occupies a capacity of roughly 57 KB's in external storage with about 1700 lines of source element.

3.4.4. REPORT

The program has two options for listing of the solutions for each program. Each program generates a solution with an extension added at the end of the product name.OPALB and HALB adds an extansion of ".OU1" and ".OU3" respectively. For example if the product name is "21-TASK" then the output files are "21-TASK.OU1" and "21-TASK.OU3" respectively.The program lists the solution;

(i) To the console,

(ii) to the printer.

Program occupies a capacity of roughly 6 KB's in external storage with about 250 lines of source element.

3.5. Computational Results

Results show that the method proposed improves the computatyional efficiency.In order to illustrate this OPALB is run for problems solved by Assche et. al. [24] and LBHA-1. [3] The solution results are given in table two for comparison purposes. Comparison is not made on the basis of computational times, but rather on the number of nodes generated. Otherwise,It becomes very difficult or even impossible to conpensate for the differences of hardware used by different researchers when testing different algorithms.

As it is seen from table 2, the proposed B&B algorithm reduces the number of nodes generated.It always gives the optimal solution.

Table 2. Computational results for different methods.

			OPALB		Assche et al.		LBHA-1
Problem Source	No.of Opr.	Cycle Time	No.of Nodes	No.of Stn's	No.of Nodes	No.of Stn's	No.of Stn's
Jackson	11	8	47	7	NA	NA	7
[13]		9	1	6	NA	NA	6
		10	5	5	10	5	5
		12	10	4	9	4	4
		17	3	3	NA	NA	3
		24	5	2	NA	NA	2
Tonge	21	18	2	6	20	6	6
[25]		19	2	6	12	6	6
		20	2	6	19	6	6
		21	12	5	15	5	5
Tonge	70	346	167	11	1279	11	NA
[3]		349	171	11	618	11	NA
		352	176	11	657	11	NA
		355	185	11	2887	11	NA
		358	186	11	881	11	NA

NA - Not available.

IV. APPLICATION OF THE PROPOSED METHOD TO MIXED MODEL ASSEMBLY LINE BALANCING

The main objective in mixed-model line balancing is to assign work elements to stations in a manner that each station has an equal amount of work load on shift basis. Because station by station assignments on individual models may lead to an uneven flow of work along the line for any given model.

Thus, it is necesary to modify the conventional single model assembly line balancing methods for this purpose. In order to avoid the uneven flow of work on the line,

(i) The compound precedence concept has been introduced. Compound precedence relationships are obtained by superimposing the precedence relations of each model. The assignments are made then according to the resultant compound precedence matrix.

(ii) Concept of the aggregate task times is another extension. he aggregate task time for a task is defined as the total time spent on that task during a whole shift.

Thomopoulos [14] showed that with some modification single model assembly line balancing methods are applicable to mixed-model assemly line balancing. For that, mixed-model problem is reduced to single model case by taking cycle time

as the shift time, precedence relationships as compound precedence and the task times as aggregate task times.

MIXALB computer package makes all of the above modiications. The program mainly uses the same algorithm, namely OPALB's algorithm. Its input module is different and is called MIXINP. The solution algorithm is as follows:

Step 1 . Get input for each model seperately - MIXINP.

Step 2 . a. Prepare compound precedence.

b. Calculate aggregate element times.

c. Set cycle time as shift time.

Step 3 . Run OPALB.

Step 4 . a. Calculate the smoothing factor in model base.

b. Save the solution into a text ile and STOP.

MIXALB is run for several example problems with satisfactory results.

V. CONCLUSION

In this Study, B&B method is studied for single and mixed model assembly lines. In order to make the problem more realistic additional restrictions are added to the regular precedence and cycle time restrictions. An upper bound concept which is used for feasibility check of lower bounds is introduced. This increases the computational efficiency of B&B method proposed. As it is seen in 3.5, the algorithm always obtains an optimal solution while generating less number of nodes compared to the B&B method proposed by Assche and Herroelen [13].

The next step of this study should be one of the main topics of production planning in the future. Adoption of the software to the robotics assembly line. There are two planning phases for a robotic line:

(i) Initial planning : This is deciding on the number of robots given the available number of robots, their abilities and speeds. This planning action is to obtain the production rates, namely cycle time given the number of robots that are decided to be equipped. A robot can do certain types of operations. If number of operations that a robot can do increases, then its price increases consequently. That is why, there is a trade-off between having a multipurpose robot or more than one robot that can do different operations.

(ii) Ongoing planning : This planning action is to decide on the assignments for an existing line. It aims at obtaining minumum number of robots being used on the line given the production rate. Capabilities and speeds of the robots are restrictions of the problem. Speeds of robots are set in a range. So that, in the case of idle time after balancing the line acording to the highest speeds of the robots, speeds of partly idle robots can be decreased without violating cycle time which might lead to energy saving and decrease in breakdown probability.

For both problems, the software can be of much help with some small modifications in the structure of the solution method.

BIBLIOGRAPHY

1. Geoffrey Boothroyd, Corrado Poli, Laurence E. Murch Automatic Assembly, Marcel Dekker Inc., New York and Basśel, 1982.

2. Salveson, M.E., "The Assembly Line Balancing Problem," The Journal of Industrial Engineering, 3, 18-25, 1955.

3. Baybars, Ilker, "An Efficient Heuristic Method for the Simple Assembly Line Balancing Problem," Int. J. Prod. Res. , 24, 149-166, 1986.

4. Baybars, Ilker, "A Survey of Exact Algorithms," Management Science, 31, 909-931, 1986.

5. Patterson, James H., and Albracht, Joseph J., "Assembly-Line Balancing: Zero-One Programming with Fibonacci Search," Operations Research, 23, 166-172, 1975.

6. Thangavelu, S. R. and Shetty, C.M., "Assembly Line Balancing by Zero-One Integer Programming," AIIE Transactions, 3, 61-68, 1971.

7. Kilbridge, M. D., and Wester, L., "A Review of Analtical Systems of Line Balancing," Operations Research, 10, 626-638, 1962.

8. Dar-El, E. M. "Solving Large Single-Model Assembly Line Balancing Problems - A Comparative Study," AIIE Transactions, 7, 302-310, 1975.

9. Dar-El, E. M. , "MALB - A Heuristic Technique for Balancing Large Single-Model Assembly Lines," AIIE Transactions, 5, 343-356, 1973.

10. Dar-El, E.M. and Rubinovitch, Y. , "MUST-A Multiple Solutions Technique for Balancing Single-Model Assembly Lines," Management Science, 25, 1105-1114, 1979.

11. Pinto, Peter, Dannenbring, David G. , and Khumawala, Basheer M., "A heuristic Network Procedure for the Assembly Line Balancing Problem," NRLQ, 25, 299-307, 1978.

12. Fisher, Robert P., and Falkner, Charles H. ," An Assessment of the Applicability of Current balancing Techniques to Robotic Assembly Lines," Research Report, Dept. of Ind. Eng., University of Wisconsin-Madison, 1985.

13. Assche, F. Van and Herroelen, W. S. , " An Optimal Procedure for the Single-Model Deterministic Assembly Line Balancing Problem," European Journal of Operations Research, 3, 142-149, 1978.

14. Thomopoulos, Nick T., "Mixed-Model Line Balancing with Smoothed Station Assignments," Management Science, 16, 693-703, 1970.

15. Thomopoulos, Nick T., "Line Balancing Sequencing for Mixed-Model Assembly ," Management Science, 14, B59-B75, 1967.

16. Pinto, Peter, Dannenbring, David G. , and Khumawala, Basheer M. , "Branch and Bound and Heuristic Procedures for Assembly Line Balancing with Paralleling Stations," Int. J. Prod. Res. , 19, 565-576, 1981.

17. Pinto, Peter, Dannenbring, David G., and Khumawala, Basheer, M. , "A Branch and Bound Algorithm for Assembly Line Balancing with Paralleling," Int. J. Prod. Res. , 13, 183-196, 1965.

18. Buxey, G. M. ,"Assembly Line Balancing with Multiple Stations," Management Science, 20, 1010-1021, 1974.

19. Kao, Edward P. C. , "Computational Experience with a Stochastic Assembly Line Balancing Algorithm," Comput. & Ops. Res. , 6, 79-86, 1979.

20. Sphicas, Georghios P., and Silverman, Fred P., "Deterministic Equivalents for Stochastic Assembly Line Balancing," AIIE Transactions, 8, 280-282, 1976.

21. Johnson, Lynwood A., Montgomery, Douglas C., Operations Research in Production Planning, Scheduling, and Inventory Control, John Wiley & Sons Inc., New York, 1974.

22. Schofield, Norman A., "Assembly Line Balancing and the Application of Computer Techniques-NULISP," Comput. and Indus. Engng., 3, 53-69, 1979.

23. Bullinger, H. J.,Schad, G., Lentes, H. P."Interactive Assembly Line Balancing-INTAKT," Toward the Factory of the Future, 773-777, 1985.

24. Alkan, Orhan, "Montaj Hatları Optimum Dengeleme Metodu: Bir Fabrika Örneği," 1. Otomotiv ve Yan Sanayii Sempozyumu, 4-8 Kasım, 1985.

25. Tonge, Fred M. "Summary of a Heuristic Line Balancing Procedure," Management Science, 21, 21-39, 1960.