In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

7/25/68

# ASSEMBLY LINE BALANCING BY ZERO-ONE INTEGER PROGRAMMING

A THESIS

Presented to

The Faculty of the Graduate Division

by

S. R. Thangavelu

In Partial Fullfillment

of the Requirements for the Degree

Master of Science in the School of Industrial Engineering

Georgia Institute of Technology

December, 1969

ASSEMBLY LINE BALANCING BY

ZERO-ONE INTEGER PROGRAMMING

Approved:

_____

Chairman

_____

_____

Date approved by Chairman: Dec. 15, 1969

# ACKNOWLEDGMENTS

The author is deeply indebted to Dr. C. M. Shetty, his thesis advisor, for his excellent guidance and encouragement during the course of this study.

The author is grateful to Dr. D. E. Fyffe and Dr. G. E. Unger for their participation in the reading committee.

TABLE OF CONTENTS


| | Page |
|---|---|
| ACKNOWLEDGMENTS | ii |
| LIST OF TABLES | iv |
| LIST OF ILLUSTRATIONS | v |
| SUMMARY | vi |
| CHAPTER | |
| I. INTRODUCTION | 1 |
| II. LITERATURE SURVEY | 5 |
| III. ZERO-ONE INTEGER PROGRAMMING FORMULATION OF ASSEMBLY LINE BALANCING PROBLEM | 12 |
| Formulation of Bowman and White | |
| Modification of Above Formulation | |
| Reduction in the Number of Constraints and Variables | |
| Comparison with the Other Formulation of Bowman | |
| IV. SOLUTION PROCEDURE | 21 |
| Modifications in Geoffrion's Algorithm for Assembly Line Balancing Problem: | |
| Handling Equality Constraints | |
| Simplified Infeasibility Checks | |
| Augmentation | |
| Estimation of Lower Bound | |
| Surrogate Constraints | |
| V. COMPUTATIONAL RESULTS AND RECOMMENDATIONS | 42 |
| APPENDICES | 50 |
| BIBLIOGRAPHY | 84 |

# LIST OF TABLES


| | | Page |
|---|---|---|
| Table I. | Coefficients $b_k$ and $a_{kij}$ for the Integer Programming Formulation of Assembly Line Balancing Problem. . . . . | 29 |
| Table II. | Computational Results Using Proposed Algorithm . . . | 44 |
| Table III. | Calculation of Number of Variables Required for Assembly Line Balancing Problem by Proposed Method . . . . . | 54 |

# LIST OF ILLUSTRATIONS


| Figure | | Page |
|---|---|---|
| 1. | Flow Diagram of Geoffrion's Algorithm . . . . . | 24 |
| 2. | Flow Diagram for the Proposed Algorithm . . . . . | 37 |
| 3. | Precedence Diagram for 8 Element Problem . . . . . | 50 |
| 4. | Precedence Diagram for 11 Element Problem . . . . . | 53 |

# SUMMARY

An assembly line balancing problem is to minimize the number of stations along an assembly line to perform a given set of elements without exceeding the cycle time for the station and satisfying the precedence relations between the elements. Several approaches have been proposed to solve the above problem. The objective of this thesis is to develop a computationally efficient integer programming method for the assembly line balancing problems.

In this report the zero-one integer programming formulation of Bowman and White is modified to reduce the number of constraints and variables. Geoffrion's integer programming algorithm is then applied to the revised formulation. This formulation has a special structure which will permit the elimination of certain steps and simplify some others if augmentation is done in a particular manner.

A computer program based on the proposed algorithm has been tested for a number of problems up to 45 elements. The results indicate that the proposed algorithm is considerably faster than the method of Held et al. which is considered the best method at present among the exact methods. The proposed method also reduces computer storage requirements.

# CHAPTER I

## INTRODUCTION

Although assembly line methods have been used in industry from the turn of the century, the first published article on the assembly line balancing problem, by Salveson (31), appeared only in 1953. In an assembly line balancing problem, a set of elements, their performance times, the precedence relations between them and the cycle time for the work stations are given. The problem is to assign the elements to a sequence of work-stations such that the required numbers of workstations are minimized. The assembly line balancing problem has been attacked by different approaches. In this thesis we present an efficient method of solving assembly line balancing problems by implicit enumeration approach to integer programming.

We will now give a general description of the assembly line balancing problem and its importance. The following assumptions are made in formulating this problem.

(i) Elements: It is a rational division of the total work content in an assembly process and is further indivisible for the problem considered. The performance time of an element is constant. The performance time of a set of elements is given by the sum of performance times of individual elements belonging to the set.

(ii) Serial line: Each assembly product is processed at the workstations in a definite order and no two workstations operate on the

same product simultaneously. Thus the total assembly line is considered to be serial with no "feeder" or "parallel" assembly lines.

(iii) Precedence Relations: All restrictions on the order of execution of elements may be expressed by precedence relations of the form "element i" must precede "element j" for which we use the notation $J_i << J_j$. The elements are numbered in such a way that if $J_i << J_j$, then $i < j$.

(iv) Cycle Time: The interval of time between the completion of products is a constant C called the cycle time. $C = W/D$ where D is the demand rate and W is the actual working time in which D units are required.

(v) Station: The location on the assembly line where a given amount of work is performed by an operator is called a station. The sum of performance times of elements assigned to a station must be less than or equal to the cycle time. All the stations are identical and any element can be a candidate for a station provided it satisfies the precedence requirements. In other words we do not consider zoning constraints which restrict each element to be performed in certain work stations only.

Following Gutjahr and Nemhauser (13) we define the assembly line balancing problem as follows: Given a finite set of elements $A = \{J_1,....J_N\}$ and their associated performance times $t_1,t_2,...t_N$, the cycle time C and a partial ordering << defined on A, find a sequence of subsets $(A_1,A_2,....A_M)$ satisfying the following conditions:

(i) $\bigcup_{i=1}^{i=M} A_i = A$,

(ii) $A_i \cap A_j = \phi$ (the empty set), $i \neq j$,

(iii) $t(A_i) = \Sigma_{J_j \epsilon A_i} t_j \leq C$, $i = 1,\ldots,M$,

(iv) if $J_k << J_m$ and $J_k \epsilon A_i$ and $J_m \epsilon A_r$, then $i \leq r$

(v) $\sum_{i=1}^{M} (C - t(A_i))$ is to be minimized.

Next we consider the industrial importance of the assembly line balancing problem. Assembly line methods are used in a large number of industries, prominent among them being the automotive, electronic and appliance industries. A survey conducted by the Illinois Institute of Technology Research Institute reveals that assembly costs amount to 26% of the total manufacturing costs in the above industries (29). Another survey conducted by Wester and Kilbridge (34).indicates that the American automotive industry wastes on the average about twenty five percent of the assemblers' time through uneven work assignments. Hence one can see the scope and necessity for using analytical methods to solve assembly line balancing problems.

A variety of analytical methods have been proposed to solve the assembly line balancing problem. These approaches are based on integer programming, dynamic programming, network, Branch and bound or heuristics. In Chapter II the important methods in each approach will be discussed.

The algorithm proposed in this thesis is based on the implicit enumeration approach to the integer programming problem. In Chapter III we will discuss the integer programming formulations of assembly line balancing problem by Bowman (5) and White (35) and the modification herein

proposed to the above to reduce the number of constraints and variables.

Chapter IV deals with the implicit enumeration approach of Geoffrion (9) to the integer programming problem and the modifications that can be made to take advantage of the special structure of the assembly line balancing problem. The computational results of the algorithm for a number of problems ranging from 7 elements to 45 elements, collected from published literature are given in Chapter V. The comparisons with other algorithms are indicated wherever such results are available.

The computer program written for UNIVAC 1108 in FORTRAN V and the specimen inputs and outputs for an assembly line balancing problem are given in the Appendix.

# CHAPTER II

## LITERATURE SURVEY

The various methods for solving the assembly line balancing problem can be grouped under the following categories: (i) exact methods which assure an optimum solution in all cases (ii) approximate methods which may yield an optimum solution in some cases but whose main objective is to treat large sized problems with special constraints. The approaches of integer programming, dynamic programming network methods and branch and bound methods are exact methods where as a number of heuristic techniques fall in the category of approximate methods. The integer programming formulation suggested in this thesis is an exact method and it will be more appropriate to make comparisons with other exact methods.

Hence in this chapter while discussing various methods we devote greater attention to the exact methods and in particular to the computationally efficient dynamic programming approach of Held et al. (14). Of the approximate methods we discuss the sequencing method of Arcus (1) which yields near optimum solutions in many cases with less computing time than other methods. The ranked positional weight technique of Helgeson and Birnie is also outlined here since we use it to generate a good starting solution in the method suggested in this thesis. Only the methodology of various approaches is discussed in this chapter. The computational aspects of various methods and a comparison of their computing times are presented in Chapter V.

## Exact Methods

We first discuss some of the exact methods of solving a line balancing problem.

### Integer Programming

Integer programming formulations have been given by Salveson (31) and Bowman (5). In Salveson's formulation, the variables represent the various combinations of elements to form a station. Since all combinations are required to be created at the start of the procedure itself, the number of variables become too large and hence is not considered practical for problems of even modest size.

Bowman (5) has suggested two integer programming formulations. Both the formulations are discussed in Chapter III. His first formulation is modified in this thesis to reduce the number of constraints and is found to be better compared to the second formulation.

### Dynamic Programming

Jackson (19) proposes a dynamic programming algorithm in which the stations are the stages, the elements are the stage variables and the objective is to reduce the number of stations which are not empty. He uses certain dominance properties to reduce the number of sequences, which are more suited for hand calculations for small problems than computer solutions for moderate sized problems.

Held, Karp and Shareshian (14) offer a dynamic programming formulation which has been found to be computationally most efficient among the exact methods. To understand this method a few definitions are essential. A subset $S = \{J_{i_1}, J_{i_2}, \ldots J_{i_n}\}$ of the elements is said to be a Feasible Set if $J_j \varepsilon S$ and $J_i << J_j$ imply that $J_i \varepsilon S$. Thus the elements of

a feasible set may be performed in some order without the prior completion of any other element and without violating any precedence constraint: A Sequence (ordered set), $\sigma = (J_{i_1}, J_{i_2}, \ldots J_{i_{n(\sigma)}})$ is said to be a Feasible Sequence if for all $1 \leq q \leq n(\sigma)$, $\{J_{i_1}, J_{i_2}, \ldots J_{i_q}\}$ is a feasible set. There is a natural correspondence between feasible sequences and feasible sets defined by two mappings: if $\sigma = (J_{i_1}, J_{i_2}, \ldots J_{i_{n(\sigma)}})$ is a feasible sequence, $F(\sigma) = \{J_{i_1}, J_{i_2}, \ldots J_{i_{n(\sigma)}}\}$; if S is a feasible set, $F^{-1}(S) = \{\sigma | F(\sigma) = S\}$.

Associated with each feasible sequence is a "cost" $Z_\sigma = (r-1)C+T^{(r)}$ where r is the number of stations required to perform the elements in the given order and $T^{(r)}$ is the sum of the performance times of elements assigned to the rth station. If a feasible sequence $\sigma^*$ is formed by adjoining an element $J_\ell$ to the end of $\sigma$ then $Z_{\sigma^*} = Z_\sigma + \Delta (Z_\sigma, t_\ell)$ where

$$\Delta(Z_\sigma, t_\ell) = \begin{cases} t_\ell \text{ if } T^{(r)} + t_\ell \leq C, \\ C - T^{(r)} + t_\ell \text{ otherwise.} \end{cases}$$

Out of the number of feasible sequences that can be formed from the elements of a feasible subset the one with a minimum cost is used to represent the cost of the feasible subset, i.e.,

$$Z(S) = \min_{\substack{J_\ell \in S \\ (S - J_\ell) \text{ feasible}}} \{Z(S - J_\ell) + \Delta \, | Z(S - J_\ell), t_\ell |\}$$

where $(S - J_\ell)$ is the set obtained by deleting $J_\ell$ from S. Held et al. (14) uses the above relation recursively to determine $Z(\{J_1, J_2, \ldots J_N\})$.

The approach given above involves only feasible sets which are far less numerous than feasible sequences. Further at any time only the feasible sets and their costs are to be stored. Even then memory requirements increase with the number of elements because of the rapid increase in the number of feasible sets. Hence the authors use an approximation procedure in which they subdivide the set of N elements into small groups of elements when N is large. These groups replace the individual elements in their recursive relationship.

Network Methods

Klein (21) has shown how a line balancing problem can be formulated as an assignment problem or as a shortest path problem in a directed network when the order of operations is specified. When partial ordering only is given it is necessary to enumerate all feasible ordering of operations and apply the above method repeatedly.

Gutjahr and Nemhauser (14) formulate assembly line balancing as a single shortest path problem. In their network formulation the nodes represent feasible sets $S_i$ defined earlier. A directed arc exists from node $S_i$ to $S_j$ if $S_i \subset S_j$ and $t(S_j)-t(S_i) \leq C$ where $t(S_i) = \sum_{J_k \in S_i} t_k$, $k = 1,\ldots,N$, $i = 1,\ldots,r$, where r is the total number of feasible sets. The "distance" of arc ij is given by $C - \left[t(S_j) - t(S_i)\right]$. The solution to the assembly line balancing problem is given by the assignments along the shortest path from node $S_o$ to $S_r$ where $S_o = \phi$, i.e., a null set and $S_r = \{J_1,\ldots J_N\}$. In this problem the arc lengths are such that it is

sufficient to find any path from the origin to the destination node containing a minimal number of arcs. A procedure is also described in (13) for selecting the arcs of the above network such that a path from node $S_o$ to any other node will have the least number of arcs possible.

Branch and Bound Methods

Jaeschke (20) has proposed a branch and bound method in which the given set of elements $\{J_1,\ldots J_M\}$ are divided into a sequence of subsets $(A_1,\ldots A_k,\ldots A_M)$ subject to the conditions (i) and (v) in Chapter I. Associated with subset $A_k$ is a lower bound

$$Z_k = kC - \sum_{j_j \epsilon A_1,\ldots,A_k} t_j$$

of the objective function representing idle time. The procedure may be summarized as follows. From the starting node of the tree diagram nodes $A_{1i1},\ldots A_{1p_1}$, are created where $p_1$ represents the number of nodes at the first level and $A_{1i}$ represents a particular combination of assigning elements to the first station. Nodes at the kth level of the tree diagram represent the various combinations of assigning elements to kth station and are denoted by $A_{k1}$, $A_{k2}$, $A_{ke},\ldots A_{kp_k}$, and

$$Z_{ke} = k\,C - \sum_{J_j \epsilon A_{1i_1}, A_{2i_2},\ldots,A_{ke}} t_j \qquad (1)$$

where $A_{1i_1},\ldots A_{ke}$ represent the particular sets along the branch leading up to node $A_{ke}$. Branching is done from the node having the minimum

lower bound. A feasible solution is obtained when the subsets along a branch contain all the elements of the assembly line balancing problem.

The Branch and Bound method proposed by Martens (26) is very much similar to that of Jaeschke (20) discussed above. For node (ke) the lower bound of the objective function representing number of stations needed is estimated as

$$Z_{ke} = k + \left[ \sum_{J_j \notin A_1 i_1, A_2 i_2, \ldots, A_{ke}} t_j \Big/ C \right]^R \qquad (2)$$

where notation $\lceil g \rceil^R$ indicates the lowest integer $\geq g$. The equations (1) and (2) express lower bounds on the total idle time and number of stations respectively for a particular node. The two bounds are equivalent and hence computationally one method does not have an advantage over the other.

## Approximate Methods

We now consider a few of the approximate methods. They mainly cater to the needs of providing good approximate solutions and treating line balancing problems with special constraints such as Zoning Constraints, "two-man" tasks, variable performance times and multimodel sequencing etc. Approximate methods have been proposed by Kilbridge and Wester (21), Tonge (32), Hoffman (17), Mansoor (24), Moodie and Young (28), Arcus (1) and Helgeson and Birnie (15). The method of Helgeson and Birnie is presented here because it is used in the algorithm presented in this thesis. The method of Arcus is summarized as it has achieved

good computational efficiency. For a discussion of the other heuristic methods we refer the reader to review articles (7), (18), and (33).

In the Ranked Positional Weight method of Helgeson and Birnie, priority is given to elements on the basis of their positional weights. The positional weight of element $J_k$ is $W_k = t_k + \sum_{j \in V(k)} t_j$ where $V(k) = \{J_j | J_k << J_j\}$. The elements are ranked in the order of their positional weights, the element with the largest weight coming first. One proceeds to assign elements in the order of ranking. If an element takes longer than the time remaining in the station or violates precedence relations it is passed over and the next element is tried. This process is continued until no more elements can be assigned to the station. The computational effort in obtaining the solution is minimum and it gives fairly good solutions. Hence this method has been used to provide good initial solution in the procedure used in this thesis.

In the technique developed by Arcus (1) called COMSOAL (Computer Sequencing of Assembly Line Balancing) the essential idea is the random generation of feasible sequences. For any given partial sequence there are certain elements which can be selected next on a probabilistic basis. These probabilities are specified by several heuristic rules and are modified as the solution procedure progresses. As the sequence is being generated, elements are assigned to stations. A large number of sequences (1000 in his program) are generated in this way and those which give the fewest number of stations are chosen. Certain amount of duplication of sequences occur since many sequences yield the same combination of elements for the stations.

# CHAPTER III

## ZERO-ONE INTEGER PROGRAMMING FORMULATION OF ASSEMBLY LINE BALANCING PROBLEM

Integer Programming is one of the exact methods which has been proposed to solve the assembly line balancing problem. Bowman has given two formulations for this problem. The first formulation uses general integer variables and is discussed below. The second formulation is a mixed integer formulation and is treated later in this chapter. White (35) has modified Bowman's first formulation by using zero-one integer variables thereby reducing the number of variables and constraints. Yet Ignall (18) and several others consider these formulations as impractical even for moderate sized problems. In this chapter we present some modifications to the formulation of Bowman-White which reduces the number of constraints greatly and enables problems to be solved in computer requiring less computing time compared to other exact methods.

First we consider the Bowman-White's formulation. Let

$$x_{ij} = \begin{cases} 1 \text{ if element } J_j \text{ is assigned to station } A_i, \\ 0 \text{ otherwise,} \end{cases}$$

$t_j$ = performance time of element $J_j$,

$C$ = cycle time,

$M_o$ = minimum number of stations required given by $\left\lceil \sum_{j=1}^{N} t_j/C \right\rceil^R$,

where notation $|g|^R$ indicates lowest integer $\geq g$,

$M$ = upper limit on the number of stations,

$I(q)$ = $\{p | J_p << J_q$ and there is no element $J_k$ such that $J_p << J_k$ and $J_k << J_q\}$, i.e., $J_p$ is an immediate predecessor of $J_q$,

$H_q$ = the number of elements in $I(q)$.

$R$ = $\sum_{j=1}^{N} H_j$, i.e., the total number of immediate precedence relations.

and $F = \{j | j \notin I(q), q=1,\ldots,N\}$ i.e., set of elements having no followers.

The requirement that the total time of elements assigned to a work-station be less than the cycle time is expressed as

$$\sum_{j=1}^{N} t_j x_{ij} \leq C, \quad i=1,\ldots,M. \tag{3}$$

The constraints below assure that all the elements are performed:

$$\sum_{i=1}^{M} x_{ij} = 1, \quad j=1,\ldots,N. \tag{4}$$

The precedence relation between $J_p$ and $J_q$ where $J_p \varepsilon I(q)$ is expressed by

$$x_{kq} \leq \sum_{i=1}^{k} x_{ip}, \quad k=1,\ldots,M. \tag{5}$$

Thus to express R precedence relationships MR equations will be required. Equations (3) through (5) represent M + N + MR constraints of the problem.

The objective function (to be minimized) of this linear program takes the form:

$$\text{minimize} = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij},$$

where
$$c_{ij} = t_j \left[ \sum_{k \in F} t_k + 1 \right]^{(i-M_o-1)}, \quad i = M_o + 1, \ldots, M, \quad \text{and } j \in F, \qquad (6)$$

$$= 0 \text{ otherwise.}$$

The purpose of the above objective function is to make later stations exceedingly costly and to assign the elements to the earliest station possible on the assembly line. Stations $1, \ldots, M_o$ must certainly be used and need assume no cost. Only elements with no followers need positive costs in the objective function, i.e., they may be last on the line. The nature of cost explosion is to make one unit of a later assignment more costly than the sum of all preceeding station assignments.

It is observed from Equation (5) in order to express the precedence relationship between the two elements, as many equations as the number of stations are used. We will show below that only one equation is adequate to express the precedance relationship between two elements, greatly reducing the total number of equations required.

If $J_p \in I(q)$, then the precedence requirement can be completely specified by

$$\sum_{i=1}^{i=M} (\mu_{ip} x_{ip} - \mu_{iq} x_{iq}) \geq 0 \quad , \tag{7}$$

where $\mu_{ip}$, $\mu_{iq}$ are nonnegative numbers satisfying

$$\mu_{ip} \geq \mu_{iq} > \mu_{i+1,p} \ , \ i=1,\ldots,M, \text{ and } \mu_{M+1,p} \equiv 0 \tag{8}$$

We will now show that sets of equations (3), (4) and (7) completely specify the restrictions of the problem.

Remark 1: Given a solution $X = [x_{ij}]$, $i=1,\ldots,M$, $j=1,\ldots,N$, $x_{ij} = 0$ or 1 to equation (4), then equation (5) is satisfied iff equation (7) is satisfied.

Proof: We first prove that Equation (7) implies Equation (5). Suppose the kth equation in (5) is not satisfied, i.e., $\sum_{i=1}^{k} x_{ip} < x_{kq}$.

Noting Equation (4), this implies that $x_{ip} = 0$, $i=1,\ldots,k$, and $x_{kq} = 1$. Hence noting Equation (4) and Equation (8),

$$\sum_{i=1}^{M} (\mu_{ip} x_{ip} - \mu_{iq} x_{iq}) = \sum_{i=k+1}^{M} \mu_{ip} x_{ip} - \mu_{kq}$$

$$\leq \mu_{k+1,p} - \mu_{kq} < 0, \text{ which violates Equation (7).}$$

On the otherhand Equation (5) implies Equation (7). Suppose Eq. (7) is not satisfied. Noting (4) and (8) this implies $x_{rp} = 1$ and $x_{kq} = 1$ for a specific $r > k$. Now considering the kth equation of (5), we get

$$\sum_{i=1}^{k} x_{ip} - x_{kq} = 0 - 1 < 0, \text{ a contradiction.}$$

This completes the proof.

It may be noted that the solution to Equation (8) is not unique. A convenient set of values would be the location of the station counted from backwards i.e., $\mu_{ij} = M - i + 1$, In this case $\mu_{ij}$ is independent of the elements $J_j$ and depends only on station i.

With the modifications discussed above the total number of equations to completely state an assembly line balancing problem reduces from N + M + MR to N + M + R. If it is assumed that the average number of elements per station denoted by K remains constant even when the total number of elements increase, then M is approximately equal to N/K. The total number of immediate precedence relations is of the same order as N and can be expected to be less than 2N (and this was empirically checked for a few problems). Hence the number of constraints in the proposed formulation increases linearly with N whereas in Bowman-White formulation it increases approximately to the square of the number of elements.

Next we consider a method to reduce the number of variables. From the maximum number of stations and the precedence relations, we can determine the earliest station $E_k$ and the latest station $L_k$ for an

element $J_k$ as follows.

$$E_k = \left[ \left( \sum_{i \varepsilon P(k)} t_i + t_k \right)/C \right]^R ,$$

$$L_k = M - \left[ \left( \sum_{i \varepsilon V(k)} t_i + t_k \right)/C \right]^F ,$$

where $P(k) = \{j | J_j << J_k\}$,

$V(k) = \{j | J_k << J_j\}$,

$[g]^R$ = the lowest integer $\geq g$,

and $[g]^F$ = the highest integer $\leq g$.

Hence the number of variables required to describe the assignment of element $J_k$ to a station is $L_k - E_k + 1$, and the total number of variables required to express the problem is reduced from MN to

$$\sum_{k=1}^{N} (L_k - E_k + 1).$$

An example for the above method is presented in Appendix II. The number of variables required to describe an 11 element problem drops from 66 to 42. In general the number of variables required to describe the problem will depend upon the maximum number of stations and precedence relations.

The above method of reducing the number of variables has not been incorporated in the computer program used to solve assembly line balancing

problem. The main utility of the above method is to reduce the memory capacity rather than the overall computation time.

We will now consider the second formulation of Bowman where an assembly line balancing problem is expressed as a mixed integer programming problem. Comparison of the number of equations and variables with the formulation presented earlier will be indicated later. Let

$$T = \sum_{j=1}^{N} t_j,$$

$v_j$ = the starting time of element $J_j$, a continuous variable,

$$y_{kj} = \begin{cases} 1 \text{ if } J_k \text{ is started first,} \\ 0 \text{ if } J_j \text{ is started first,} \end{cases}$$

where $J_k$ and $J_j$ are elements such that neither $J_k << J_j$ nor $J_j << J_k$, and

$\mu_j$ = an integer variable

such that if $\mu_j = k$ then element $J_j$ is performed in the $k + 1^{th}$ station.

The precedence relations are specified by the following constraints:

$$v_i + t_i = v_j, \quad i \varepsilon I(j), \quad j = 1,\ldots,N.$$

To prevent two elements not ordered by precedence relations (i.e., neither $J_k << J_j$ nor $J_j << J_k$) from using the same clocktime, the following constraints are required.

$$(T + t_j)\, y_{kj} + (v_k - v_j) \geq t_j \ ,$$

$$(T + t_k)\, (1 - y_{kj}) + (v_j - v_k) \geq t_k \ .$$

The following set of constraints ensure that the stations are not overloaded and one element is not split between stations but assigned to one workstation only.

$$v_j + t_j \leq C(\mu_j + 1) \text{ and } v_j \geq C\mu_j, \quad j=1,\ldots,N$$

The objective function is to minimize a (continious) variable d representing the clocktime at the completion of the last element in the sequence. Clearly we must impose the following restriction on all elements without any followers which we have already denoted by a set F.

$$v_j + t_j \leq d, \quad j \varepsilon F.$$

In the above mixed integer programming formulation there are $2N + Q + 1$ variables and $R + 2Q + 2N$ constraints, where Q is the total number of unordered pairs of elements for the given problem and R is the total number of immediate precedence relationships defined earlier in this chapter.

For N elements the maximum number of unordered pairs can be $N(N - 1)/2$. Let $p = Q/\left[N(N-1)/2\right]$. In assembly line balancing problems encountered in practice the value of p ranges between 0.15 to 0.85 (25). Since Q is large compared to N, the number of variables in Bowman's second

formulation is given by $Q \approx p\ N(N-1)/2 \approx p\ N^2/2$ neglecting small order numbers. Let K be the average number of elements per station. Then $M \approx N/K$ and the number of variables (i.e. MN) in our formulation is approximately $N^2/K$. Bowman's second formulation is likely to have more variables compared to our formulation if $Kp > 2$, which is usually satisfied for most of the problems. The number of constraints in Bowman's second formulation are much larger compared to our formulation since $2Q + 2N + R > M + N + R$. For a 45 element problem (33) Bowman's second formulation requires 655 variables and 1275 constraints compared to 450 variables* and 117 constraints for the formulation proposed in this thesis.

---

*This does not consider the reduction in number of variables by the method suggested earlier in this chapter.

# CHAPTER IV

## SOLUTION PROCEDURE

In the previous chapter an improved zero-one integer programming formulation has been suggested for an assembly line balancing problem. There are a number of algorithms reviewed in (12) for solving zero-one integer programming problems. The algorithm used in this thesis follows basically the Balas' additive algorithm as modified by Geoffrion (9). In this chapter we summarize the above algorithm and discuss the various simplifications made in its various steps to take advantage of the special structure of the problem and thereby reduce computation time.

A zero-one integer programming problem can be written in the form

$$\min \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad b_k + \sum_{j=1}^{n} a_{kj} x_j \geq 0, \quad k=1,\ldots,m, \qquad \text{(P)}$$

$$x_j = 0 \text{ or } 1, \quad j=1,\ldots n.$$

To understand the Balas' algorithm certain definitions are useful. A <u>partial solution</u> S is defined as an assignment of binary values to a subset of the variables. Any variables not assigned a value are called <u>free</u>. An element j(-j) of S indicates that $x_j$ takes on the value 1(0)

in the partial solution. The order in which the elements of S are written is used to represent the order in which the elements have been generated. A completion of a partial solution S is defined as a solution that is determined by S together with a binary specification of the values of the free variables. An element belonging to S may or may not be underlined. The significance of an underline at the $k^{th}$ position (counting from left) is that all completions of the partial solution up to and including the $k^{th}$ element complemented (i.e., multiplied by -1) have been accounted for. A partial solution is said to be augmented when a free variable is assigned a fixed value.

For a given partial solution it may be possible to find a feasible completion that minimizes the objective function among all feasible completions. Secondly it may be possible to determine that the partial solution has no feasible completion better than the current best known feasible solution. In either case we say that the partial solution has been fathomed. Once a partial solution is fathomed, the implicit enumeration scheme backtracks to a partial solution which will allow further augmentation. When this occurs attempt is made to fathom the new partial solution.

The feasibility tests usually include tests for binary infeasibility and conditional binary infeasibility. A constraint is said to be binary infeasible if it has no binary solution and is said to be conditionally binary infeasible if its binary feasibility is contingent upon certain of the variables taking on particular binary values. It is easily verified that $\beta + \Sigma_j \alpha_j x_j \geq 0 (>0)$ is binary infeasible iff $\beta + \Sigma_j \max \{0, \alpha_j\} < 0 (\leq 0)$; and $\beta + \Sigma_j \max \{0, \alpha_j\} - |\alpha_{j_0}| < 0 (\leq 0)$ implies

$x_j = 0$ or 1 according as $\alpha_{j\delta} < 0$ or $\alpha_{j\delta} > 0$, in any binary solution of $\beta + \Sigma_j \alpha_j x_j \geq 0$ (>0).

Associated with any partial solution S is an integer program ($P_S$) involving the free variables only.

$$\text{Min} \quad Z^S + \sum_{j \notin S} c_j x_j \qquad (P_S)$$

$$\text{subject to} \quad b_k^S + \sum_{j \notin S} a_{kj} x_j \geq 0, \quad x_j = 0 \text{ or } 1$$

where notation $j \in S (\notin S)$ refers to the fixed (free) variables,

$$Z^S = \sum_{j \in S} c_j x_j \text{ and } b_k^S = b_k + \sum_{j \in S} a_{kj} x_j \ .$$

The flow chart for Geoffrion's algorithm is given below which is basically the diagram given in (9) except that step 2a has been added. This step corresponds to the conditional binary infeasibility tests where certain free variables are required to be zero or one to ensure feasibility.

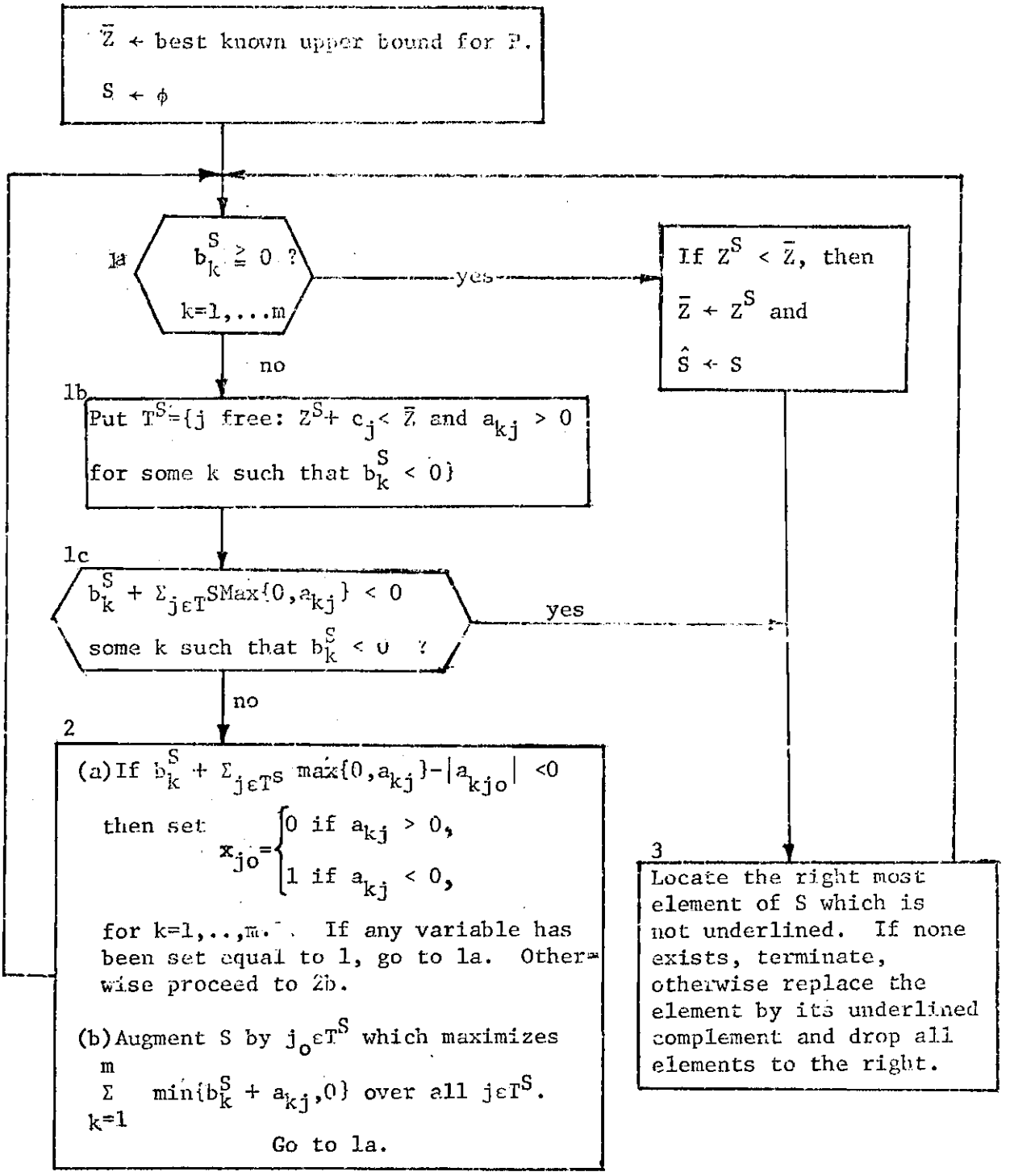


Figure 1. Flowchart of Geoffrion's Algorithm

It may be observed that the above algorithm requires that the constraints be in inequality form whereas the formulation discussed in Chapter III involves equality constraints. Furthermore the problem has certain special structure which can be taken advantage of. After a discussion of these aspects below we will then develop a simplified version of Geoffrion's algorithm to solve an assembly line balancing problem.

In the formulation presented in Chapter III, Equation (4), i.e., $\sum_{i=1}^{M} x_{ij} = 1$, $j = 1,\ldots,N$, are equality type constraints. Balas (2) suggests replacement of equality constraints by inequality constraints giving Equations (9) and (10) below in place of (4).

$$1 - \sum_{i=1}^{M} x_{ij} \geq 0 \quad , \quad j=1,\ldots,N. \tag{9}$$

and

$$1 - \sum_{i=1}^{M} x_{ij} \leq 0 \quad , \quad j=1,\ldots,N. \tag{10}$$

The following remark enables us to substitute the constraints in (10) by a single constraint

$$N - \sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij} \leq 0 \ . \tag{11}$$

Remark 2: Given $X = \left[x_{ij}\right]$, $x_{ij} = 0$ or 1, $i=1,\ldots,M$, $j=1,\ldots,N$, satisfying Equation (9), then Equation (10) is satisfied iff Equation (11) is satisfied.

Proof: We first prove that Equation (11) implies Equation (10). Suppose Equation (10) is not satisfied. Then

$$1 - \sum_{i=1}^{M} x_{ij} > 0$$

for some $j = j_o$. From Equation (9), we have

$$\sum_{j=j_o} (1 - \sum_{i=1}^{M} x_{ij}) \geq 0 \quad .$$

On adding,

$$N - \sum_{j=1}^{N} \sum_{i=1}^{M} x_{ij} > 0$$

which contradicts Equation (11). Hence Equation (11) implies Equation (10). On the otherhand adding Equations in (10), we find that Equation (10) implies Equation (11).

Thus the following constraints fully describe the assembly line balancing problem.

$$\text{Min} \sum_{i=M_o+1}^{M} \sum_{j \in F} x_{ij} t_j \left[ \sum_{k \in F} t_k + 1 \right]^{(i-M_o-1)} \qquad (12)$$

subject to

$$C - \sum_{j=1}^{N} t_j x_{ij} \geq 0 \quad , \qquad i=1,\ldots,M, \tag{13}$$

$$1 - \sum_{i=1}^{M} x_{ij} \geq 0, \qquad j=1,\ldots,N, \tag{14}$$

$$\sum_{i=1}^{M} (M-i+1)(x_{ip} - x_{iq}) \geq 0, \qquad p \in I(q) \quad q=1,\ldots,N, \tag{15}$$

and

$$-N + \sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij} \geq 0. \tag{16}$$

Letting $m = M + N + R + 1$ the above equations can be written as,

$$\text{Min} \quad \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij}$$

$$\text{s.t. } b_k + \sum_{i=1}^{M} \sum_{j=1}^{N} a_{kij} x_{ij} \geq 0, \quad k=1,\ldots,m.$$

In the above equations, k=1,...,M, refer to (13), k=M + 1,...,M+N refer to (14) and k = M + N + 1,..., M + N + R refer to (15) and k = m, refers to Equation (16).

We notice that the matrices $\left[b_k\right]$ and $\left[a_{kij}\right]$ have special structure

in the integer programming formulation of the assembly line balancing problem. This is summarized in Table I. The cost coefficients of the variables are such that only few of them are positive and the rest are all zero. Now we will discuss the modifications of Geoffrion's algorithm to take advantage of the special structure.

Given a partial solution S, we have already defined

$$b_k^S = b_k + \sum_{(i,j)\in S} a_{kij}x_{ij} \quad ,$$

so that we must have

$$b_k^S + \sum_{(i,j)\notin S} a_{kij}x_{ij} \geq 0.$$

Under the augmentation procedure adopted (discussed in detail later) we would like to show that if $b_k^S + a_{kij} < 0$ for some $k = 1,\ldots, m-1$, then $x_{ij}$ must be equal to zero for feasibility.

For showing this we first prove the following remark. The terms partial solution and completion used in this remark have already been defined earlier in this chapter.

Remark 3: Consider Equations (17) and (18) below in binary variables $x_{ij}$ ,

$$1 - \sum_{i=1}^{M} x_{ij} \geq 0 \quad , \qquad j=1,\ldots N, \tag{17}$$

| Equation k | $b_k$ | $a_{kij}$ , i=1,...,M. j=1,...,N. |
|---|---|---|
| 1.....M | C>0 | $a_{kij} = \begin{cases} 0 \text{ if } i \neq k, \text{ all } j \\ < 0 \quad \text{otherwise.} \end{cases}$ |
| M+1....M+N | 1 | $a_{kij} = \begin{cases} -1, \ j = k - M, \text{ all } i \\ 0 \quad \text{otherwise} \end{cases}$ |
| M+N+1,...M+N+R | 0 | $a_{kij} = \begin{cases} M-i+1, & j = p < q \\ -(M-i+1), & j = q \\ 0 & j \neq p,q \end{cases}$ Hence $a_{kip} = -a_{kiq} > a_{ki+1p} = -a_{k,i+1,q}$ |
| m | -N | $a_{mij} = 1$, all i,j. |

Table I. Coefficients $b_k$ and $a_{kij}$ for the Integer Programming Formulation of Assembly Line Balancing Problem.

$$\sum_{i=1}^{M} \sum_{j=1}^{N} a_{kij} x_{ij} \geq 0 , \tag{18}$$

where

$$a_{kij} = \begin{cases} (M - i + 1) > 0, & j=p , \\ 0 , & j \neq p,q , \\ -(M - i + 1) < 0, & j = q . \end{cases} \tag{19}$$

Let S define a partial solution which does not violate Equations (17) or (18) and let

$$b_k^S = \sum_{(i,j) \in S} a_{kij} x_{ij} . \tag{20}$$

If for some $(i_o, q) \notin S$,

$$\text{(i)} \quad b_k^S + a_{ki_oq} < 0 , \tag{21}$$

$$\text{(ii)} \quad x_{ip} \neq 1 \text{ for all } i > i_o , \tag{22}$$

$$\text{(iii)} \quad (i,j) \in S \text{ for } i \leq i_o - 1 \text{ and all } j , \tag{23}$$

then $x_{i_oq} = 0$ in any completion of S satisfying Equations (17) and (18).

Proof: Suppose $x_{iq} = 1$ for some $i \neq i_o$. Then obviously $x_{i_o q} = 0$ to satisfy Equation (17). Hence we need to consider only cases where $x_{iq} \neq 1$ for all $i$. Note that in general

$$x_{ij} \neq 1 \quad \text{implies} \quad x_{ij} = 0, \quad \text{if } (i,j) \in S \cdot \tag{24}$$

Now, since $x_{iq} \neq 1$, noting Equation (19), we get from Equation (20),

$$b_k^S = \sum_{(i,p)\in S} a_{kip}\, x_{ip} \cdot$$

But from Equation (21) and (19), $b_k^S < - a_{ki_o q} = a_{ki_o p}$. Hence $x_{ip} \neq 1$ for $i \leq i_o$ since Equation (19) implies $a_{kip} > a_{k,i+1,p}$. But $x_{ip} \neq 1$, for $i > i_o$ from Equation (22). Hence $x_{ip} \neq 1$, i.e., $x_{ip} = 0$, $(i,p) \in S$ from Equation (24) giving $b_k^S = o$.

Now suppose $x_{i_o q} = 1$. Then obviously we do not have a feasible completion if

$$b_k^S + a_{ki_o q} + \max \left[ \sum_{\substack{(i,j)\in S \\ (i,j)\neq i_o q}} a_{kij} x_{ij} \right] < 0 \cdot \tag{25}$$

But $b_k^S = 0$ and $a_{ki_o q} = - (M - i_o + 1)$. Further from Equation (19), $a_{kij} > 0$ only for $j = p$, and $x_{ip} = 0$, $i \leq i_o$ from Equation (23) and our proof that $x_{ip} = 0$, $(i,p) \in S$. Hence noting Equation (17), the last term

in Equation (25) is less than or equal to $a_{k,i_o+1,p} = (M-i_o)$. Hence inequality in (25) is satisfied and hence we must have $x_{i_oq} = 0$. This complets the proof.

Our solution procedure discussed later ensures that the hypothesis of Remark 3 is satisfied. Hence $b_k^S + a_{kij} < 0$ for some k = M+N+1,..., M+N+R, implies $x_{ij} = 0$. Furthermore the solution procedure ensures $b_k^S \geq 0$, k=1,...,m=1, at every stage and $a_{kij} \leq 0$, k=1,...,M+N, i=1,...,M and j=1,...N. Obviously then if $b_k^S + a_{kij} < 0$, k=1,...,M+N, we must have $x_{ij} = 0$, since otherwise there is no way to get a feasible completion.

It may be noted that in equations k = M+1,...,M+N, if some $x_{i_oj} = 1$, then $x_{ij} = 0$ for all $i \neq i_o$. Since this rule of augmentation is very simple and augments several variables at a time these constraints have been treated logically rather than use the conditional binary infeasibility tests. Incidentlly this reduced the storage requirements for solving the problem by about 25%.

It may be further noted that for some k, if $b_k^S \geq 0$, then $b_k^S + a_{kij} < 0$, only if $a_{kij} < 0$. From Table I, $a_{kij} < 0$ only for i = k in the set of equations from k = 1,...,M, and for $H_q$ equations among k=M+N+1,...,M+N+R, where $H_q$ is the number of elements which immediately precedes any given element q and

$$R = \sum_{q=1}^{N} H_q$$

as defined earlier in Chapter III.

We will now show that certain other steps in Geoffrion's algorithm shown in flow diagram 1 can be eliminated or simplified. In step 1a, since we always maintain $b_k^S \geq 0$, k=1,...,m-1, only the last equation needs to be tested. Step 1b is designed to select variables for which $Z^S + c_{ij} < \bar{Z}$ and $a_{kij} > 0$ for some k with $b_k^S < 0$. In our algorithm $b_k^S < 0$ for only k = m and $a_{mij} = 1$ for all i and j. Hence the selection of variables reduces to testing $Z^S + c_{ij} < \bar{Z}$. Again in step 1c, usually

$$\sum_{(i,j)\notin S} a_{mij} > b_m^S$$

(except when a large number of $x_{ij}$ are set equal to zero in Step 3b and 3c of the proposed algorithm discussed later). Hence the test is seldom effective and is not included in our algorithm. As discussed earlier we have simplified the conditional binary infeasibility test given in step 2a.

Now we consider step 2b of Geoffrion's algorithm. We have $b_k^S + a_{kij} \geq 0$, k=1,...,m-1, for all $i,j \notin S$ since by the conditional binary infeasibility test, we have set already $x_{ij} = 0$, for variables which violated the above condition. For k=m, $b_m^S + a_{mij} = b_m^S + 1$ for all $(i,j) \notin S$. Thus the test fails to discriminate among variables in the extent by which infeasibility is reduced. Hence the test is not used in our proposed algorithm.

In addition to the simplifications discussed above in the various steps of Geoffrion's algorithm an additional step for calculating a

lower bound on the objective function of the feasible completion is added in the proposed algorithm. This is obtained by first finding the revised estimate of the minimum number of stations needed, denoted by $i_2$. The lower bound is then computed from

$$Z_{est} = Z^S + c_{i_2 j_f} \quad ,$$

where $t_{j_f} = \min_{j \in F} t_j$, i.e., $J_{j_f}$ is the element without followers and with the least performance time. This will enable us to fathom a partial solution even when a few variables have been assigned binary values.

We will now describe the algorithm. Initially $b_k^S \geq 0$, $k=1,\ldots,m-1$, and $b_m^S < 0$. The following sequence $\sigma$ is used in the augmentation steps 1 and 3a below.

$$\sigma:(x_{11}, x_{12},\ldots x_{1N}; x_{21},\ldots,x_{2N};\ldots;x_{M1},\ldots,x_{MN})$$

Procedure:

Step 1: Start from an initial feasible partial solution $S_o$ and $\bar{Z}$ obtained by a heuristic method satisfying Equations (22) and (23) of Remark 3 where $(i_o q)$ is the left most free variable in sequence $\sigma$. Go to step 7. (Otherwise one can start from $S_o = \phi$, $\bar{Z} = \sum_i \sum_j c_{ij}$. In such a case, set $i_o = 1$, $j_o = 1$ and proceed to step 3b.)

Step 2: If $b_m^S \geq 0$, go to step 6. Otherwise go to step 3a.

Step 3a: Select the left most free variable in sequence $\sigma$, say $x_{i_1 j_1}$. If $i_1 = i_o$, set $j_o = j_1$ and proceed to step 3b. If $i_1 > i_o$ proceed to step 4.

Step 3b: If $Z^S + c_{i_o j_o} > \bar{Z}$ or $b_k^S + a_{k i_o j_o} < 0$ for some $k=1,\ldots,m-1$, then

set $x_{i_o j_o} = 0$ and return to 3a. (Note: As discussed earlier only $1+H_q$ equations have $a_{k i_o j_o} < 0$ and need to be checked.) Otherwise proceed to step 3c.

Step 3c: Set $x_{i_o j_o} = 1$ and $x_{i j_o} = 0$, for all $i \neq i_o$ and $(i, j_o) \notin S$. Return to step 3a.

Step 4: Estimate the minimum number of stations $i_2$ given by

$$i_2 = i_o + \left[ \sum_{j \in P} t_j / C \right]^R$$

where

$$P = \{j | x_{ij} \neq 1, \text{ for any } i\} \quad .$$

Estimate the lower bound $Z_{est} = Z^S + c_{i_2 j_f}$ and $t_{j_f} = \min_{j \in F} t_j$. Go to step 5.

Step 5: If $Z_{est} \geq \bar{Z}$, then go to step 7. Otherwise set $i_o = i_1$ and return to step 2.

Step 6: If $Z^S \leq \bar{Z}$, then set $\bar{Z} = Z^S$ and store the corresponding partial solution as $\hat{S}$. Go to step 7.

Step 7: Locate the rightmost element $x_{i_1 j_1}$ of S which is not underlined. If none exists terminate. Otherwise replace the element by the underlined complement and drop all other elements to the right. Set $i_o = i_1$ and return to step 2.

The above steps are represented in a flow diagram shown in Figure 2. In step 1, of the procedure, an initial feasible solution is obtained by a simple heuristic method of Helgeson and Birnie (15) described in Chapter II. Their technique is executed basically with the same steps as described above except that the element numbers are replaced by the

ranks of the elements which are based on their positional weights. Since we do not have a good bound $\bar{Z}$ at the beginning we omit steps 4 and 5 while executing the above method. The steps in the above heuristic method are such that the partial solution representing the initial feasible solution satisfies the conditions Equations (22) and (23) of Remark 3 where we let $(i_o q)$ to be the left most free variable in sequence $\sigma$. We have not represented the details of the heuristic method in the flow chart given in Figure 2.

Block 8 in the flow diagram represents the reading of data for assembly line balancing problem. Block 9 represents the calculations made to get the coefficients $b_k$, $c_{ij}$, and $a_{kij}$ for the integer programming problem.

We now consider the usefulness of surrogate constraints to the integer programming formulation of the assembly line balancing problem. A surrogate constraint is obtained by a nonnegative linear combination of the original constraints plus the constraint $(\bar{Z} - \sum_{(i,j)} c_{ij}x_{ij})$ where $\bar{Z}$ is the value of currently best known feasible solution of (P). More precisely each surrogate constraint has the form $U(b + AX) + (\bar{Z} - \sum_{(i,j)} c_{ij}x_{ij}) > 0$ for some nonnegative m-vector U. The strength of a surrogate constraint is related to its ability to fathom a partial solution represented by conditions 1b and 1c of the flow diagram given in Figure 1.

To obtain the "strongest" surrogate constraint Geoffrion (10) solves the following linear program.

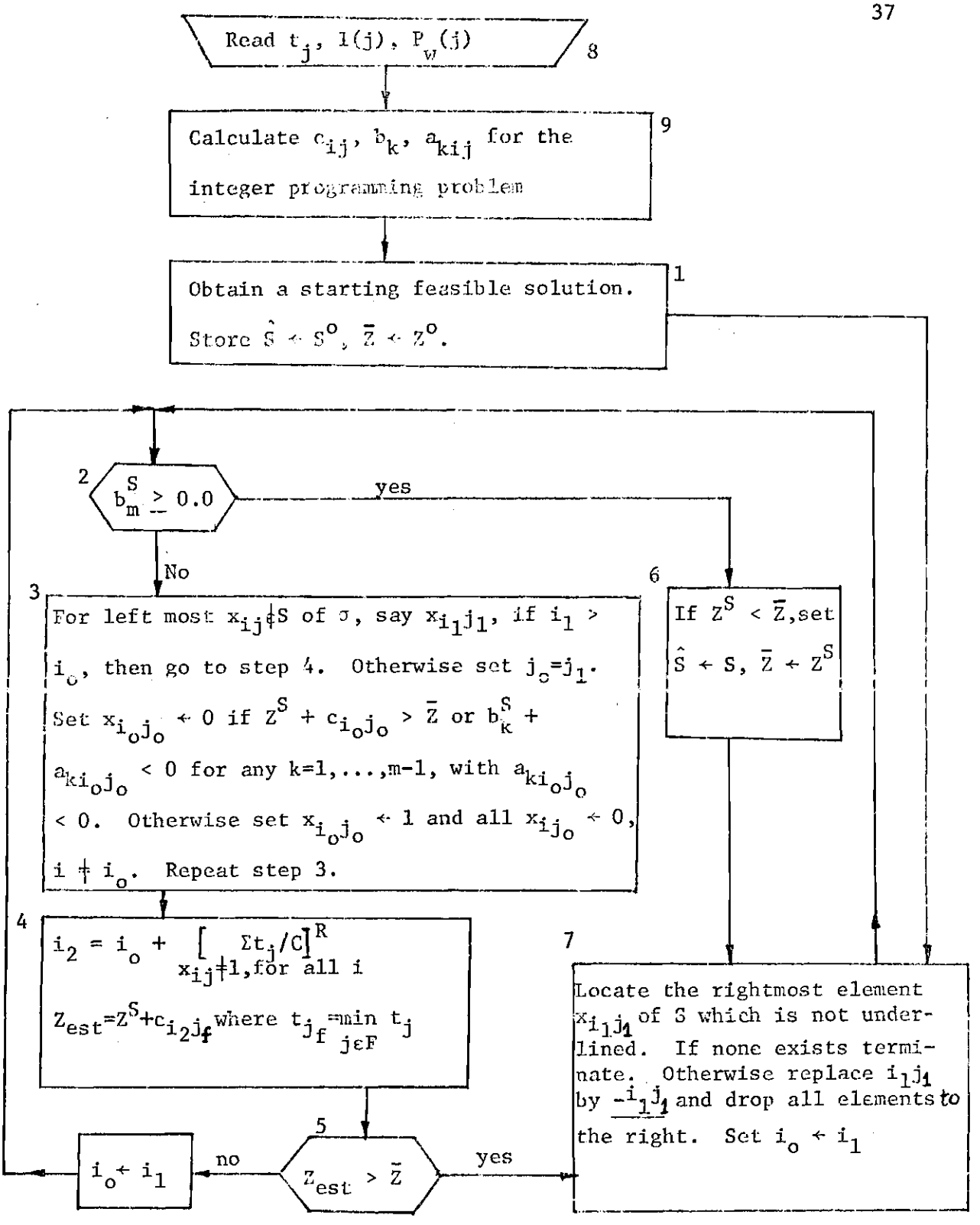


Figure 2. Flowchart for the Proposed Algorithm

$$\underset{u,w}{\text{Min}} \quad \sum_{k=1}^{m} u_k b_k^S + \bar{Z} - Z^S + \sum_{(i,j)\notin S} w_{ij}$$

$$\text{Subject to} \quad w_{ij} \geq \sum_{k=1}^{k=m} u_k a_{kij} - c_{ij}, \; ij \notin S, \qquad (LP_S)$$

$$w_{ij} \geq 0, \; ij \notin S, \; u_k \geq 0, \; k=1,\ldots,m,$$

where $w_{ij}$ are the dual variables of the constraint $x_{ij} \leq 1$ in the continuous version of the problem (P).

We examine the Kuhn-Tucker equations for optimality to the Problem $(LP_S)$ and show under what conditions in assembly line balancing problem we do not obtain a surrogate constraint.

In $(LP_S)$, since $\bar{Z} - Z^S$ is a constant as far as u and w are concerned it can be omitted. Hence the Lagrangian function for $(LP_S)$ becomes

$$F(u,w,\lambda) = \sum_{k=1}^{k=m} u_k b_k^S + \sum_{ij\notin S} w_{ij} - \sum_{ij\notin S} \lambda_{ij} \left(w_{ij} - \sum_{k=1}^{m} u_k a_{kij} + c_{ij}\right)$$

where $\lambda_{ij}$ are the Lagrangian multipliers.

The Kuhn-Tucker conditions are,

$$b_k^S + \sum_{ij \notin S} \lambda_{ij}(a_{kij}) \geq 0, \quad k=1,\ldots,m, \tag{26}$$

$$1 - \lambda_{ij} \geq 0, \qquad ij \notin S, \tag{27}$$

$$u_k \left[ b_k^S + \sum_{ij \notin S} \lambda_{ij} a_{kij} \right] = 0, \quad k=1,\ldots,m, \tag{28}$$

$$w_{ij} \left[ 1 - \lambda_{ij} \right] = 0, \qquad ij \notin S, \tag{29}$$

$$w_{ij} - \sum_{k=1}^{m} u_k a_{kij} + c_{ij} \geq 0, \quad ij \notin S, \tag{30}$$

$$\lambda_{ij} \left[ w_{ij} - \sum_{k=1}^{m} u_k a_{kij} + c_{ij} \right] = 0, \quad ij \notin S, \tag{31}$$

$$u_k \geq 0, \; k=1,\ldots m, \; w_{ij} \geq 0, \; \lambda_{ij} \geq 0, \; ij \notin S.$$

The conditions (26),(27) are exactly the same as the conditions of the problem $(P_S)$ in which $\lambda_{ij}$ replaces $x_{ij}$. There is one to one correspondence between $\lambda_{ij}$ and $x_{ij}$ and hence in further discussion we will use $x_{ij}$ instead of $\lambda_{ij}$ in the above equations.

Let $x = \left[x_{ij}\right]$, $i=1,\ldots M$, $j=1,\ldots N$ be a solution to Equations (26)

and (27). $u_k = 0$, k=1,...m, $w_{ij} = 0$, $ij \notin S$, satisfies Equations (28), (29) and (30) since $c_{ij} \geq 0$ (Equation (12), Chapter IV). It is also a solution to equation (31) if

$$c_{ij}x_{ij} = 0, \quad i=1,...M, \quad j=1,...N \cdot \tag{32}$$

From Equation (12) of Chapter IV we find that $c_{ij} > 0$ only for $i > M_o$ and $j \varepsilon F$ where $M_o$ is the minimum number of stations required and F is the set of elements with no followers. The number of variables whose $c_{ij} > 0$ forms a small proportion of the total number of variables and hence in many cases the solutions to the LP problem would satisfy Equation (32). In such cases since $u_k = 0$, k=1,...M we obtain $(\bar{Z} - CX) > 0$ as a surrogate constraint.

Also the optimalvalue of the objective function $v(LP_S)$ serves as an useful lower bound for the partial solution S in other problems. The lower bound calculated in step 4 of the procedure described earlier is at least as good or better than $v(LP_S)$ since we allow fractional solutions to $LP_S$ but not in step 4. of the procedure. The bound calculated by step 4 takes very little calculation effort unlike the LP solution.

For a few problems tested, the results point to the same conclusion obtained above. For 4 problems (up to 11 elements) we used the above program with and without surrogate constraints calculated by linear programming. The results are presented in Appendix III. The computation time without LP is much less than the computation time with LP. We

obtained ($\bar{Z} - CX$) as the surrogate constraint in all the iterations it was evaluated.

The program with LP* also required too much storage to solve assembly line balancing problems with more than 15 elements. Even eliminating the LP subroutine was inadequate from the point of view of obtaining good computational results comparable to other algorithms for assembly line balancing problems. Hence the above modifications suggested in this chapter were incorporated. The computational results obtained with the modified program are presented in the next chapter.

*The memory capacity in number of words required by Geoffrion's algorithm (8) is approximately $n^2 + n(3m + 18) + 9000$.

# CHAPTER V

## COMPUTATIONAL RESULTS AND CONCLUSIONS

In this chapter we consider the computational results of the procedure discussed in Chapter IV. A computer program based on the above procedure has been tested for a number of problems up to 45 elements. The computing time for those problems are presented here and compared mainly with other exact methods. Also we compare the various methods on the basis of their solution technique, and storage capacity needed. The results presented here indicate the proposed algorithm compares favorably with other exact methods in computing time and storage requirements.

The computer program was written in FORTRAN V for UNIVAC 1108 computer with 60,000 word capacity. It is based on (8) though it has been modified considerably as indicated in Chapter IV. The input to the computer program for assembly line balancing problem includes the performance times, immediate predecessors, the total number of immediate predecessors and the positional weight of each element. Instead of using the conventional precedence matrix which itself requires a large storage capacity, the immediate predecessors of each element are identified by a small matrix as suggested by Moodie and Young (28). The output gives the list of elements assigned to various stations and the calculation time for intermediate steps. The computer program can handle assembly line balancing problems for which $M \cdot N \leq 450$ and

M+R+1 $\leq$ 75*. The computer program together with sample input and output are given in Appendix IV.

The computational results of this algorithm for 6 different problems taken from various articles are given in Table II. The results show that smaller problems up to 30 elements require computing time of less than a second. The 45 element problem of Kilbridge and Wester (33) is solved in about 5 seconds of computer time. Bigger problems could not be solved because storage requirements exceed the computer core memory capacity+ available.

We will now discuss the reduction of computing time obtained by the modifications suggested to Geoffrion's algorithm (9) in Chapter IV. From Appendix III we find that for 7 element and 11 element problem (cycle times = 10) the computing times for Geoffrion's algorithm without LP subroutine was 0.152 and 3.758 secs respectively. From Table II we find that the computing times for the above problems by the proposed algorithm is reduced to 0.026 and 0.853 secs. Hence one can see that computing times have been reduced by a factor of about 4 by the modifications.

Direct comparison with other techniques is beset with number of difficulties. The computation times depend upon the computer,

*The storage word capacity required for a general assembly line balancing problem is approximately M·N (M+R+12) + (N+Q) + 9000, where Q = $\text{Max}H_q$, q=1,...N, i.e., the maximum number of direct predecessors for any element.

+Auxiliary storage devices like magnetic tapes can be used for large problems but comparisons of computing time with other algorithms becomes difficult.

| Problem No. | No. of Elements | Cycle Time | Max No. of Stations | Min No. of Stations (line length) | No. of Prec. Relations | No. of Constraints | No. of Variables | Time for Generation of Coeff. | Time for Initial Soln. by Approx. Method | Time at Last Feas. Solution | Time for Backtracking | Total Computation Time | No. of Feasible Sets* | Order Strength+ | Problem Reference |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 10 | 4 | 3 | 5 | 10 | 28 | 0.015 | 0.005 | 0.006 | 0.004 | 0.026 | 22 | 0.53 | (26) |
| 2 | 11 | 10 | 6 | 5 | 13 | 20 | 66 | 0.018 | 0.008 | 0.824 | 0.011 | 0.853 | 50 | 0.58 | (18) |
| | 11 | 12 | 6 | 4 | 13 | 20 | 66 | 0.018 | 0.008 | 0.013 | 0.019 | 0.049 | 50 | 0.58 | (18) |
| 3 | 19 | 12 | 9 | 8 | 29 | 39 | 171 | 0.025 | 0.015 | 0.016 | 0.030 | 0.071 | 1986 | 0.48 | (17) |
| 4 | 21 | 14 | 10 | 8 | 27 | 38 | 210 | 0.024 | 0.017 | 0.018 | 0.036 | 0.078 | 187 | 0.71 | (32) |
| | 21 | 18 | 9 | 6 | 27 | 37 | 189 | 0.023 | 0.016 | 0.072 | 0.051 | 0.146 | 187 | 0.71 | (32) |
| | 21 | 20 | 8 | 6 | 27 | 36 | 168 | 0.023 | 0.016 | 0.017 | 0.030 | 0.070 | 187 | 0.71 | (32) |
| | 21 | 21 | 8 | 5 | 27 | 36 | 168 | 0.022 | 0.015 | 0.094 | 0.037 | 0.153 | 187 | 0.71 | (32) |
| 5 | 28 | 1.72 | 8 | 6 | 40 | 49 | 224 | 0.026 | 0.020 | 0.104 | 0.102 | 0.232 | 326,494 | 0.24 | (16) |
| 6 | 45 | 69 | 10 | 8 | 62 | 73 | 450 | 0.100 | 0.049 | 4.661 | 0.188 | 4,949 | ** | 0.43 | (33) |

* - Found by methods described by Held et al. (14).
\+ - Explained later in this chapter.
** - Too big to calculate exactly and greater than $5 \times 10^5$

Table II. Computational Results Using Proposed Algorithm

programming language, availability and use of peripheral devices, proficiency of programmers, etc. Also we could not obtain test problems for which computation times for all other methods are available. Hence we make an indirect comparison with various other methods particularly the exact methods citing computation times for the problems solved using their algorithms.

Held et al. (14) report of a 36 element problem* with 13000 feasible sets which has been solved in 20 seconds in IBM 7090 computer** by their dynamic programming method without any approximation. Computer program based upon their successive approximation method has solved the 21 element problem mentioned earlier in an average time of 0.6 seconds in IBM 7094**, and the 45 element of problem mentioned earlier in 60 seconds of IBM 7090 (14). It must be pointed out that their program is written partly in Machine language and partly in FORTRAN.

Gutjahr and Nemhauser (13) using their network method solve a 14 element problem with 710 feasible subsets approximately in 3 secs. and 17 element problem with 6320 feasible subsets in 6 minutes in IBM 7090 using FAP language. The authors remark that their computer program has not been optimized. Jackson's dynamic programming algorithm solves 20

---

* Unfortunately the 36 element problem was not available for a comparative study.

**Auerbach Corporation's comparison data for various computers indicates that UNIVAC 1108 is about 7 times faster compared to IBM 7090 and 3 times compared to IBM 7094.

element problems in about 20 seconds in IBM 7094 using FORTRAN IV (25). Mertens (26) reports that his branch and bound algorithm takes less time compared to Held, et al, for problems less than 10 elements but his computing times are not available. He also reports that his algorithm requires too much storage capacity for problems beyond 30 elements. The same can be said for the branch and bound method of Jaeschke (20) though no computational experience has been reported. The approaches of Salveson, Bowman and Klein have not been computationally tested but one can see that they will not be comparable to any of the above exact methods.

In the approximate methods, Arcus (1) has solved the 45 element problem mentioned earlier in approximately 32 seconds of IBM 7090 computer. The other approximate methods (16), (17), (28), and (32) take considerably more time for some of the problems given in Table II for which they have been applied.

Comparing with other methods, it can be concluded that for problems up to 45 elements the algorithm presented here is faster by about 50% when compared to that of Held et al.(14) and is much faster compared to the other exact and approximate methods except to that of Arcus (1).

Now the various basic approaches to the assembly line balancing problem will be discussed. An assembly line balancing problem is combinatorial in nature and as the number of elements increases the number of feasible sequences increases rapidly. Ignall (18) indicates that a rough estimate of the number of feasible sequences is given by $N!/2^R$ where R is the total number of immediate (or covering) precedence relationships.

The actual number of feasible sequences in any problem is determined by the strength of partial ordering. A quantitative measure of order strength* is given by the ratio of actual number of immediate and implied precedence relations to the maximum number of precedence relations possible. The later is given by N(N-1)/2. The number of feasible sequences are inversely related to order strength.

As discussed in Chapter II Held et al.(14) made use of feasible sets instead of feasible sequences for storing the cost values of the solutions. The feasible sets are less numerous compared to feasible sequences. But even the number of feasible sets increases rapidly with the number of elements. A procedure for calculating the number of feasible sets for a given assembly line balancing problem is given in (14). For a 28 element problem given in (16) the number of feasible sets is about $3 \times 10^5$. Gutjahr and Nemhauser (13) also use feasible sets in their method.

Branch and bound methods (20) and (36) and the dynamic programming approach of Jackson (19) examine the various combinations of elements that can be assigned to a station. Held et al.(14) compared their method with Jackson's method (19) and stated that Jackson's method generally requires more computational effort than their method using feasible sets though for small problems Jackson's method is better than that of Held et al. Mertens (26) reports that for problems up to 10 elements the branch and bound method requires less computing time.

*It is given by (1-p) where p is as defined in Chapter III.

compared to that required by Held et al.

In solving the assembly line balancing problem by the algorithm given in Chapter IV, the elements of the partial solution S to the integer program are arranged in the order in which they have been generated. Hence this approach is equivalent to implicitly enumerating the feasible sequences. Though the method uses feasible sequences directly a number of feasible sequences are implicitly enumerated by the estimation of a good lower bound.

It appears that in assembly line balancing problem a large percentage of computing time is spent not in calculations but in the way the information is stored and retrieved as reported by Held et al.(14) and Gutjahr and Nemhauser (13). The method suggested by Geoffrion (9) and adopted in our computer program for storing S seems to be very efficient in remembering the history of enumeration. This could be one of the factors in making the computation time of our method less than that required for other algorithms.

There are a few parameters of the assembly line balancing problem which influence the computing time for an assembly line balancing problem. The important parameters are (i) number of elements (ii) minimum number of stations (iii) order strength. As expected, we can see from Table II that the computing time increases rapidly as the number of elements are increased as is true with all other techniques. From the results presented in Table II, it would be difficult to judge the effect of other two factors. A systematic study of a large number of problems similar to that done by Mastor (25) is needed to isolate the effect of various other factors.

For the algorithm suggested in this thesis, the storage requirements increase approximately to the third power* of the number of elements assuming that the numbers of elements assigned per station is about the same. Also the storage requirement is independent of the partial ordering of the elements. In other words the storage requirement will not vary with the number of feasible sequences, though during the solution procedure we implicitly enumerate the feasible sequences. This is in contrast to the methods of Held et al. (14) and Gutjahr and Nemhauser (13). The storage requirement for those methods is dependent on the number of feasible sets which increase as the order strength decreases (for a given number of elements). As Table II indicates the number of feasible sets increases much faster than to the third power of the number of elements. Also one is able to estimate in advance the storage space required much more easily than compared to other methods.

There seem to be further possibilities for improvement in the computer program particularly in the area of backtracking. Also a suitable choice of programming language will be beneficial in reducing computing time and storage. An approximate solution procedure for handling larger problems by decomposition could be attempted. It may also be possible to extend the integer programming formulation for handling special constraints like zoning constraints and multimodel sequencing.

*Storage capacity in our algorithm is mainly decided by matrix $[a_{kij}]$. The no. of elements in the matrix is approximately given by $(M+R+1)(MN)$ which is $\alpha N^3$.

# APPENDIX I-A

## EXAMPLE FOR INTEGER PROGRAMMING FORMULATION OF BOWMAN AND WHITE

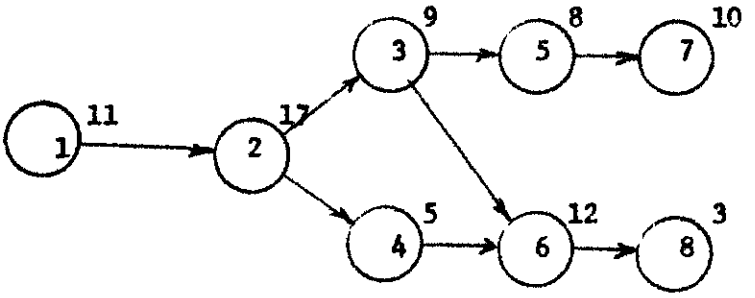


Figure 3. Precedence Diagram For 8 Element Problem Taken From Bowman (5).

The numbers inside the circles denote the element numbers and the values above the circle denote the performance times of the elements.

Cycle time = 20 units

Minimum number of stations $= \left[ \sum_{j=1}^{N} t_j / C \right]^R$

$= \left[ 75/20 \right]^R = 4$

Maximum number of stations = 7

(by a rough estimate)

Constraints

$$
\begin{aligned}
&11x_{11} + 17x_{12} + 9x_{13} + \ldots\ldots + 3x_{18} \leq 20 \\
&11x_{21} + 17x_{22} + 9x_{23} + \ldots\ldots + 3x_{28} \leq 20 \\
&- - - - - - - - - - - - - - - - - - - - - - \\
&11x_{71} + 17x_{72} + 9x_{73} + \ldots\ldots + 3x_{78} \leq 20
\end{aligned}
\qquad (33)
$$

$$
\begin{aligned}
&x_{11} + x_{21} + x_{31} + \ldots\ldots + x_{71} = 1 \\
&x_{12} + x_{22} + x_{32} + \ldots\ldots + x_{72} = 1 \\
&- - - - - - - - - - - - - - - - - \\
&x_{18} + x_{28} + x_{38} + \ldots\ldots + x_{78} = 1
\end{aligned}
\qquad (34)
$$

$$
\begin{aligned}
&x_{11} \geq x_{12};\ x_{11} + x_{21} \geq x_{22};\ x_{11} + x_{21} + x_{31} \geq x_{32}; \ldots \\
&\ldots\ldots; x_{11} + x_{21} + x_{31} + x_{41} + x_{51} + x_{61} + x_{71} \geq x_{72}; \\
&x_{12} \geq x_{13};\ x_{12} + x_{22} \geq x_{23};\ - - - - - - - - - - - \ldots \\
&\ldots\ldots; x_{12} + x_{22} + x_{32} + x_{42} + x_{52} + x_{62} + x_{72} \geq x_{73}; \\
&- - - - - - - - - - - - - - - - - - - - - - - - - - - \\
&- - - - - - - - - - - - - - - - - - - - - - - - - - - \\
&\ldots\ldots\ x_{16} + x_{26} + \ldots\ldots + x_{76} \geq x_{78}.
\end{aligned}
\qquad (35)
$$

Min $\quad 1(10x_{57} + 3x_{58}) + 14\ (10x_{67} + 3x_{68}) + 196\ (10x_{77} + 3x_{78})$

No. of constraints = 7 + 8 + (8-1) x 7 = 64

No. of variables = 56.

## APPENDIX I-B

CONSTRAINTS FOR 8 ELEMENT PROBLEM FOR THE PROPOSED FORMULATION

**Equations (33) and (34) of the previous formulation remain unchanged. The precedence relations represented by Equation (35) are modified as follows.**

$$7x_{11} + 6x_{21} + 5x_{31} + 4x_{41} + 3x_{51} + 2x_{61} + x_{71}$$

$$- (7x_{12} + 6x_{22} + 5x_{32} + 4x_{42} + 3x_{52} + 2x_{62} + x_{72}) \geq 0$$

$$7x_{12} + 6x_{22} + 5x_{32} + 4x_{42} + 3x_{52} + 2x_{62} + x_{72}$$

$$- (7x_{13} + 6x_{23} + 5x_{33} + 4x_{43} + 3x_{53} + 2x_{63} + x_{73}) \geq 0$$

- - - - - - - - - - - - - - - - - - - - - - - -

$$7x_{16} + 6x_{26} + 5x_{36} + 4x_{46} + 3x_{56} + 2x_{66} + x_{76}$$

$$- (7x_{18} + 6x_{28} + 5x_{38} + 4x_{48} + 3x_{58} + 2x_{68} + x_{78}) \geq 0$$

Hence the total number of constraints are

$$= 7 + 8 + 8 = 23 .$$

Number of variables are same as above.

# APPENDIX II

## NUMBER OF VARIABLES BY PROPOSED METHOD

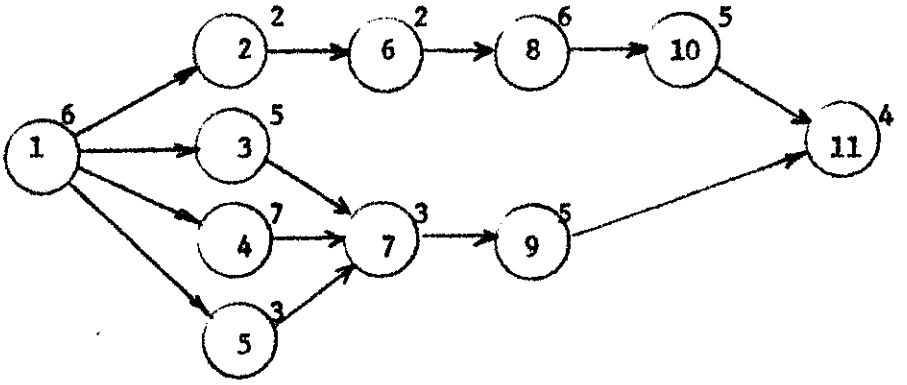


Figure 4. Precedence Diagram for 11 Element Problem from Ignall (18).

The numbers inside the circle denote the element numbers and the values above denote the elemental times.

| | |
|---|---|
| Cycle time | = 10 units. |
| Maximum No. of stations | = 6 |

| Element Number (1) | $\Sigma t_i$ $i \epsilon P(k)$ (2) | (2)+$t_k$ (3) | $E_k = \lceil (3)/C \rceil^R$ (4) | $\Sigma t_i$ $i \epsilon V(k)$ (5) | (5)+$t_k$ (6) | $L_k = M - \lceil (6)/C \rceil^F$ (7) | $L_k - E_k + 1$ (8) |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 6 | 1 | 40 | 46 | 2 | 2 |
| 2 | 6 | 8 | 1 | 17 | 19 | 5 | 5 |
| 3 | 6 | 11 | 2 | 12 | 17 | 5 | 4 |
| 4 | 6 | 13 | 2 | 12 | 19 | 5 | 4 |
| 5 | 6 | 7 | 1 | 12 | 13 | 5 | 5 |
| 6 | 8 | 10 | 1 | 15 | 17 | 5 | 5 |
| 7 | 19 | 22 | 3 | 9 | 12 | 5 | 3 |
| 8 | 10 | 16 | 2 | 9 | 15 | 5 | 4 |
| 9 | 22 | 27 | 3 | 4 | 9 | 6 | 4 |
| 10 | 16 | 21 | 3 | 4 | 9 | 6 | 4 |
| 11 | 42 | 46 | 5 | 0 | 4 | 6 | 2 |
| | | | | | | Total = | 42 |

Table III: Calculation of Number of Variables Required for Assembly Line Balancing Problem by Proposed Method.

The total number of variables required to describe the problem by the proposed modification = 42

The total number of variables required by White's formulation i.e. (MxN) = 11x6 = 66.

Reduction in the number of variables = 24.

# APPENDIX III

## RESULTS OF ASSEMBLY LINE BALANCING PROBLEM USING GEOFFRION's COMPUTER PROGRAM FOR ZERO-ONE INTEGER PROGRAMMING

| No. of Elements N | Cycle Time C | No. of Stations M | No. of Variables n | No. of Const. m | Without LP | | With LP (once in 8 iterations) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Computing Time in Seconds* | No. of Iterations | Computing Time in Seconds* | No. of Iterations |
| 5 | 10 | 3 | 15 | 14 | 0.078 | 11 | 0.106 | 15 |
| 7 | 10 | 4 | 28 | 18 | 0.152 | 29 | 0.964 | 75 |
| 9 | 10 | 4 | 28 | 25 | 0.715 | 63 | *** | |
| 11 | 10 | 6 | 36 | 31 | 3.758 | 223 | *** | |

* not including time for generation of coefficients

** exceeded the maximum time limit of 500 seconds

# APPENDIX IV

## COMPUTER PROGRAM

The code used to obtain the computational results reported in Chapter V is presented here. The present form of the code has been obtained by modifying the code for zero-one integer programming by Geoffrion (8) incorporating the various modifications discussed in Chapter IV.

The program is currently dimensioned to use 60,000 words of core in such a way that the following limits must be observed:

$$M \cdot N \leq 450$$

$$M + R + 1 \leq 75$$

If the program is to be redimensioned for any reason, such as the availability of additional core, it will be useful to know that the number of words required is approximately

$$M \cdot N (M + R + 12) + (N + Q) + 9000.$$

where M, N, R, Q have been defined in Chapter III.

Next we discuss the input data cards, the information required and the formats used to solve an assembly line balancing problem.

## INPUT

The following data cards appear for each problem to be run.

(a) option parameters card

(b) problem parameters card

(c) element data cards

(d) last elements card

(e) blank card.

Problems can be stacked by repeating cards a through e.

### Option Parameters Card

The input option parameters are:

KENUM — When intermediate output is used (NOP = 0), the fraction of all $2^n$ possible solutions that have been implicitly enumerated is printed out every KENUM times that backtracking occurs. KENUM = 20 is reasonable.

MAXT — Terminates calculations after MAXT seconds.

NOP — If equal 1, intermediate output will be suppressed; if equal 0, intermediate output will appear. Normally NOP will be set at 1.

H1,H2, — Arbitrary problem identifiers.

The fields and formats of the parameter card are as follows.

| Parameter: | Column: | Format: |
|---|---|---|
| KENUM | 1-5 | Integer |
| MAXT | 6-10 | Integer |
| NOP | 11-15 | Integer |
| H1 | 18-23 | Hollerith |
| H2 | 24-29 | Hollerith |

Problem Parameters Card

The problem parameters are:

NEL Number of elements

NRM Maximum number of predecessors for any element in a given problem.

NMST Maximum number of stations.

MJNF Number of elements without any followers.

CT Cycle time.

The fields and formats of parameter card are as follows.

| Parameter: | Column: | Format: |
|---|---|---|
| NEL | 1-5 | Integer |
| NRM | 6-10 | Integer |
| NMST | 11-15 | Integer |
| MJNF | 16-20 | Integer |
| CT | 20-28 | Floating point |

Element Data Cards

The cards for the elements should be arranged in the same serial order as the numbering of the elements. One card is required to give the details for one element. The data given for each element are

| | |
|---|---|
| J | Element number |
| TEL(J) | Performance time of element J |
| PWT(J) | Positional weight of element J, calculated by method described in (15) from the Precedence relationships (the program given can be modified to do these simple calculations). |
| NPR(J) | The elements which immediately precede the given element J. |
| NPR(NRM1) | The total number of immediate predecessors for the element J. |

The fields and formats of the above parameters are as follows:

| Parameter: | Column: | Format: |
|---|---|---|
| J | 1-3 | Integer |
| TEL(J) | 4-10 | Floating point |
| PWT(J) | 11-18 | Floating point |

For NPR(J), the element numbers are represented using I3 FORMAT i.e., 19-21, 22-24,...,etc. until there are no non-zero values. NPR(NRM1) is given using I3 format in columns ranging from 18 + NRM x 3 + 1 to 18 + NRM x 3 + 3.

Last Elements Card

The numbers of elements which do not have any followers are entered using integer format I5, i.e., in, 1-5, 6-10,...,etc.

## OUTPUT

The preliminary, intermediate and final outputs are as follows.

All the input data are printed in the output.

If NOP = 0 intermediate output is produced to reveal the data generated for the integer programming problem, the initial solution generated by the ranked positional weight method, and the course of the calculations, each feasible solution found, and a summary of progress to date after each KENUM "backtrackings", Since this information is likely to be of little incremental value to the user over the final input information, no detailed explanation is given here.

The final output gives the following: the problem designation; the message "implicit enumeration complete" or "time exceeded" according to whether termination did or did not occur within MAXT seconds; the total execution time for 0-1 integer programming problem in seconds; the objective function value; some statistical information on the course of the algorithm, such as the number of feasible solutions found, the number of iterations and the time at which last feasible solution was found; the final solution to the assembly line balancing problem giving the station numbers and the elements assigned to each station; the time taken for generation of coefficients, the time for obtaining initial solution and the total time for the solution

beginning with the assembly line balancing problem data to the final solution.

The liberally commented program together with the input and output data for the 45 element assembly line balancing problem taken from (33) are given below.

```
-RUN ASLBM,53S800012,THANGAVELU,10
-FOR,IS MAIN
C ASSEMBLY LINE BALANCING BY ZERO-ONE INTEGER PROGRAMMING
      DIMENSION A( 75,460),NPR(75,15)
      DIMENSION B(460),C(460),BS(460) S(460),SB(460),NS(460)
      DIMENSION SMAX(460),SMAXB(460),T(460)
      DIMENSION SIB(460),SIMAXB(460)
      DIMENSION TEL(100) ,JNF(10)
      DIMENSION COST(15)
      DIMENSION NSG(100),NRPT(100),RANK(100),PST(100)
      INTEGER RANK
      INTEGER S,SMAX,SC,T
      INTEGER SIB,SIMAXB
      DATA BCIB/6H1      /
      DATA BLANK/6H      /
      EQUIVALENCE (A0,A0) , (IFRST,IF
      CALL TIME (0,IT8)
C  INITIALISATION OF MEMORY TO ZERO A  D SETTING INITIAL VALUES TO PARAMETERS
  100 DO 110 I=1,460
      B(I)=0.0
      C(I)=0.0
      BS(I)=0.0
      S(I)=0
      SB(I)=BLANK
      SIB(I) =0
      NS(I)=0
      SMAX(I)=0
      SMAXB(I)=BLANK
      SIMAXB(I)=0
      T(I)=0
      DO 110 J=1, 75
      A(J,I)=0.0
  110 CONTINUE
      DO 140 J=1,100
```

```
      NSG(J)=0
      RANK(J)=0
  140 CONTINUE
      NCON=0
      NRED=0
      NAUG=0
      NOPT=0
      NID=0
      NAP=0
      NLPF=0
      NSIMP=0
      NPATH=0
      NENUM=0
      NTCE=0
      NCID2=0
      JPOST=1
      IFRST=1
      IINS=5
      MAUG =1
      IIJ1 =0
      IIJT =0
      TSUM=0.0
      ICONT =1
      ZKBAR =0.0
      READ (5,9000) KENUM,MAXT, NOP, H1,H2
 9000 FORMAT (3I5,2X,2A6)
      WRITE (6,9993)
      WRITE (6,9000) KENUM,MAXT,NOP, H1,H2
C   READ ASSEMBLY LINE DATA , NO OF ELEMENTS, MAX.NO. OF PREDECESSORS
C   FOR AN ELEMENT, MAX. NO. OF STATIONS, NO OF ELEMENTS WITHOUT FOLLOWERS,
C   CYCLE TIME
      READ (5,4001) NEL, NPN,MMST,XJUNK-CT
 4001 FORMAT (4I5,F8.3)
      WRITE (6,4001) NEL,NPN,MMST,XJUNK-CT
```

```
      WRITE (6,4002) NEL, NMST, CT
 4001 FORMAT(21H0NO. OF ELEMENTS      ,I5/
     1    21H MAX. NO. OF STATIONS,I5/
     2    11H CYCLE TIME , 10X,F8.3)
      NRTO =0
      NRM1 = NRM + 1
      NRPT(1)=0
      WRITE (6,9992)
C  READ DATA FOR ELEMENTS
C CARDS SHOULD BE INTHE PROPER SEQUENCE OF THE ELEMENTS
C ELEMENT NO IS NOT REQD BY PROGRAM FOR CALCULATIONS
 C DATA READ ARE ELEMENT NO, PERFORMANCE TIME, POSITIONAL WEIGHT,
C IMMEDIATE PREDECESSOPS, AND NUMBER OF IMMEDIATE PREDECESSORS
      WRITE (6,4054)
 4054 FORMAT (2H J,1X,7H TEL(J),2X,7H PWT(J),5X,7H NPR(J),20X,
     1 10H NPR(NRM1))
      DO 5125 J=1,NEL
      READ (5,4052) TEL(J),PWT(J), (NPR(J,K),K=1,NRM1)
 4052 FORMAT (3X,2F8.4,1613)
      WRITE (6,4053) J,TEL(J),PWT(J), (NPR(J,K),K=1,NRM1)
 4053 FORMAT (I3,2F8.4,1613)
      NRPT(I+1)=NRPT(I)+NPR(I,NRM1)
 5125 NRTO =NRTO + NPR(I,NRM1)
      READ (5,4300) (JNF(I), I=1,MJNF)
 4300 FORMAT (515)
      WRITE(6,4300)(JNF(I),I=1,MJNF)
      WRITE (6,4055) NRTO
 4055 FORMAT (40H TOTAL NO OF PRECEDENCE RELATIONSHIPS =, I5)
      CALL TIME (0,IT9)
      ELT9 =IT9-IT8
      ELT9 =ELT9 /5000.0
      IF (NOP.EQ.0) WRITE(6,776) ELT9
  776 FORMAT(1H0, 21H INITIALISATION TIME , F8.3)
C  READING OF DATA COMPLETE
```

```
C CALCULATION OF COEFFICIENTS FOR INTEGER PROGRAM FOLLOWS
      M=NMST + NRTO +1
      M1 =NMST
      M2 =M1 + NRTO
      N = NEL * NMST
      CALL TIME (0,IT10)
      ELT10=IT10 -IT9
      ELT10=ELT10/5000.0
      IF (NOP.EQ.0) WRITE (6,778) ELT 
  778 FORMAT (7H ELT10=, F8.3)
C  COEFFICIENTS OF A(K,J) AND B(K)
      DO 5150 I =1,M1
      B(I) =CT
      DO 5150 J=1,NEL
      J1 = (I-1) *NEL + J
 5150 A(I,J1) = -TEL(J)
      CALL TIME (0,IT11)
      ELT11=IT11-IT10
      ELT11=ELT11/5000.0
      IF (NOP.EQ.0) WRITE (6,779) ELT11
  779 FORMAT (27H FIRST SET CONSTRAINTS TIME ,F8.3)
      I7 = M1
      DO 5450 I=1,NEL
      J7 =I
      DO 5450 K =1,NRM
      IF (NPR(I,K).EQ.0) GO TO 5450
      I7= I7+1
      B(I7) =0.0
      J6 = NPR(I,K)
      DO 5400 J=1,NMST
      JT6 =J6+ (J-1)*NEL
      JT7= J7 + (J-1)*NEL
      A(I7,JT6) =NMST +1 -J
 5400 A(I7,JT7) = -A(I7,JT6)
```

```
 5450 CONTINUE
      CALL TIME (0,IT12)
      ELT12= IT12-IT11
      ELT12=ELT12/5000.0
      IF (NOP.EQ.0) WRITE (6,780) ELT12
  780 FORMAT (28H SECOND SET CONSTRAINTS TIME ,F8.3)
      I7 =I7+1
C LAST CONSTRAINT
      B(I7) = -NEL
      DO 5500 J=1,N
 5500 A(I7,J) = 1.0
      CALL TIME (0,IT13)
      ELT13 =IT13-IT12
      ELT13=ELT13/5000.0
      IF (NOP.EQ.0) WRITE (6,781) ELT13
  781 FORMAT (16H LAST CONST TIME ,F8.3)
C  GENERATION OF COST COEFFICIENTS
      DO 5700 I=1,NEL
 5700 TSUM = TSUM + TEL(I)
      XMIN = TSUM / CT +0.9999
      USUM =0.0
      DO 5610 I=1,NJNF
      ITEMP = JNF(I)
 5610 USUM = USUM+TEL(ITEMP)
      USUM = USUM +1.0
      COST (1) =1.0
      DO 5750 K=XMIN,NMST
      K2 =K-XMIN +1
      DO 5720 I =1,NJNF
      JJ =K*NEL + JNF(I)
 5720 C(JJ) =COST(K2)
      IF(K.EQ.NMST) GO TO 5750
      COST(K2+1) = COST(K2)*USUM
 5750 CONTINUE
```

```
      CALL TIME (0,IT14)
      ELT14 =IT14-IT13
      ELT14=ELT14/5000.0
      IF (NOP.EQ.0) WRITE (6,782) ELT14
  782 FORMAT(16H COST COEFF TIME, F8.4)
      IF (MAXT.EQ.0) MAXT=999999
      MAXT=5000*MAXT
      ZKBAR=ZKBAR+0.99999
      WRITE (6,9010) M,N
 9010 FORMAT (3HOM=,I3,2X,2HN=,I3)
      WRITE (6,9992)
 9991 FORMAT (1H )
 9992 FORMAT (1H0)
 9993 FORMAT (1H1)
      IF (NOP.NE.0) GO TO 252
      WRITE (6,9992)
      WRITE (6,9600) (C(J),J=1,N)
      WRITE (6,9992)
      WRITE (6,9600) (B(I),I=1,M)
      WRITE (6,9992)
      DO 251 I=1,M
      WRITE (6,9600) (A(I,J),J=1,N)
      WRITE (6,9991)
  251 CONTINUE
  252 CONTINUE
      CALL TIME (0,IT15)
      ELT15=IT15-IT14
      ELT15=ELT15/5000.0
      WRITE(6,783) ELT15
      IF (NOP.EQ.0) WRITE (6,783) ELT15
  783 FORMAT (19H DATA PRINTOUT TIME ,F8.3)
      WRITE (6,9992)
 9600 FORMAT (6(2X,1PE15.8))
      IF (ZKBAR.GT.0.0) GO TO 300
```

```
      ZBAR =0.0
      DO 275 J=1,N
  275 ZBAR=ZBAR+C(J)
  300 ZS=0.0
      DO 325 I=1,M
  325 BS(I)=B(I)
      DO 330 J=1,N
  330 NS(J)=J
      ELT30 =ELT10 + ELT11+ELT12+ ELT13 + ELT14
      CALL TIME (0,IT0)
      ELT16 =IT0-IT8
      ELT16=ELT16/5000.0
      IF (NOP.EQ.0) WRITE (6,784) ELT16
  784 FORMAT(27HTOTAL DATA GENERATION TIME , F8.3)
      IT1=IT0
      IT2=IT0
      IT3=IT0
C  INITIALISATION COMPLETE NOW.  START FIRST ITERATION
C RANKING BY POSITIONAL WEIGHT
      DO 5200 I=1,NEL
      BIG=0.0
      JBIG=0
      DO 5100 J=1,NEL
      IF(NSG(J).EQ.1) GO TO 5100
      IF(PWT(J).LE.BIG) GO TO 5100
      BIG =PWT(J)
      JBIG = J
 5100 CONTINUE
      RANK(I)=JBIG
      NSG(JBIG) =1
 5200 CONTINUE
      IF (NOP.EQ.0) WRITE (6,5201)
 5201 FORMAT(5H RANK,2X,8H ELEMENT,5), 18H POSITIONAL WEIGHT  )
      DO 5250 J=1,NEL
```

```
      NSG(J)=0
      JBIG= RANK(J)
      IF (NOP.EQ.0) WRITE (6,5251) J, RANK(J), PWT(JBIG)
 5251 FORMAT(I5,3X,I5,3X,F10.4)
 5250 CONTINUE
C INITIAL SOLUTION BY RANKED POSITIONAL WEIGHT METHOD
      DO 7450 I=1,NMST
      DO 7400 J=1,NEL
      JPM =RANK(J)
      IF(NSG(JPM).EQ.1) GO TO 7400
      IF(TEL(JPM).GT.BS(I)) GO TO 74(
      KRJ1 =NMST+ NRPT(JPM)+1
      KRJ2= KRJ1+NPR(JPM,NRM1) -1
      JJ =(I-1) * NEL+JPM
      IF(NS(JJ).EQ.0) GO TO 7400
      IF (NOP.NE.0) GO TO 7340
      WRITE (6,7348) I,JPM,KRJ1,KRJ2, JJ, BS(I)
 7348 FORMAT(5I5,F10.4)
 7340 CONTINUE
      IF(KRJ2.LT.KRJ1) GO TO 7351
      DO 7350 I3=KRJ1,KRJ2
      IF(A(I3,JJ)+BS(I3).LT.0.0) GO TO 7400
 7350 CONTINUE
 7351 CONTINUE
      NSG(JPM)=1
      L=L+1
      S(L)=JJ
      NS(JJ)=0
      BS(I)=BS(I)+A(I,JJ)
      ZS=ZS+C(JJ)
 7355 CONTINUE
      DO 7360 I4=KRJ1,M
 7360 BS(I4) =BS(I4)+ A(I4,JJ)
```

```
      DO 7300 IM=1,NMST
      IMJ=(IM-1)*NEL+JPM
      IF(NS(IMJ).EQ.0) GO TO 7300
      L=L+1
      S(L) =-IMJ
      SB(L) =BCIB
      SIB(L)=1
      NS(IMJ) =0
 7300 CONTINUE
 7400 CONTINUE
 7450 CONTINUE
      CALL TIME (0,IT17)
      ELT17 = IT17-IT0
      ELT17 = ELT17/5000.0
      IF (MOP.NE.0) GO TO 1906
      WRITE (6,9500) ( S(K),SB(K) ,K= ,L )
 9500 FORMAT (14(3X,I4,A1))
 1906 CONTINUE
      IF(ZS.GE.ZBAR) GO TO 3100
      GO TO 2340
C BEGINNING OF AN ITERATION
 1910 IJK=0
C  ATTEMPT TO FATHOM PARTIAL SOLUTION BY TESTS IN STEP2
 1920 CONTINUE
      IF (ZS.GE.ZBAR) GO TO 3100
      IF (BS(M).LT.0.0) GO TO 1980
      IF(RTSUM.GT.0.0.AND.BS(M).LT.0.0) GO TO 3200
      GO TO 2340
 1980 CONTINUE
      GO TO 2400
 2340 ZBAR =ZS-ZKBAR
      DO 2350 J=1,N
 2350 SMAX(J) =S(J)
      GO TO 3300
C AUGMENTATION STEP
C AUGMENTATION UNTIL FILLING A NEW STATION ATTEMPTED
```

```
 2400 K1=0
      NAP =NAP+1
      IIJ1 =IIJ1+1
      IR1 =0
 4506 CONTINUE
      I =IIJ1
      IF (BS(I).LE.0.005) GO TO 4580
      IF(BS(I).LT.CT) GO TO 4510
      IR1=1
      IIJT =I
C  SEE IF ANY VARIABLE MUST BE ZERO
 4510 DO 4550 J=1,NEL
      IF (NSG(J).EQ.1) GO TO 4550
      JJ =(I-1)*NEL +J
      IF(NS(JJ).EQ.0) GO TO 4550
      IF (TEL(J).GT.BS(I)) GO TO 4548
      KRJ1 =NAST + NRPT(J) +1
      KRJ2 =KRJ1+ NPR(J,NRN1)-1
      DO 4520 I3 =KRJ1,KRJ2
      IF(A(I3,JJ)+BS(I3).LT.0.0) GO TO 4545
 4520 CONTINUE
      IF(ZS+C(JJ).GT.ZBAR) GO TO 4548
C AUGMENTATION WITH A VALUE OF ONE
      L=L+1
      S(L) =JJ
      NSG(J)=1
      NS(JJ)=0
      BS(I) =BS(I)+ A(I,JJ)
      ZS =ZS+C(JJ)
 4522 CONTINUE
 4525 CONTINUE
      DO 4530 I4=KRJ1,M
 4530 BS(I4) =BS(I4)+A(I4,JJ)
```

```
C  LOGICAL TREATMENT OF SECOND SET OF CONSTRAINTS ,( SETTING
C  CERTAIN VARIABLES EQUAL TO ZERO)
      DO 4540 IM=1,NMST
      IMJ = (IM-1)*NEL + J
      IF(NS(IMJ).EQ.0) GO TO 4540
      L =L+1
      S(L) =-IMJ
      SIB(L) =1
      SB(L) = BCIB
      NS(IMJ) = 0
 4540 CONTINUE
      GO TO 4550
C  SETTING CERTATAIN VARIABLES EQUAL TO ZERO BECAUSE OF CONDITIONAL
C  BINARY FEASIBILITY TESTS
 4545 L=L+1
      S(L) =-JJ
      SIB(L) =1
      SB(L)=BCIB
      NS(JJ)=0
 4550 CONTINUE
      IF(IR1.EW.1) GO TO 4555
 4560 CONTINUE
C ESTIMATE A CHEAP LOWER BOUND
C  LOWER BOUND =(ASSIGNED STATIONS) + (SUM OF TIMES OF UNASSIGNED ELEMENTS/
C                  CYCLE TIME)
 4555 CONTINUE
      RTSUM =0.0
      DO 4590 I =1,NEL
      IF(NSS(I).EQ.1) GO TO 4590
      RTSUM=RTSUM+ TEL(I)
 4590 CONTINUE
      IF(RTSUM.LE.0.0) GO TO 1910
      IREM =(RTSUM/CT) + 0.999
      MSTR= IIJ1 + IREM
```

```
      !
      IF(NSTR.GT.NMST) GO TO 3100
      JJ =(NSTR-1)*NEL +JNF(1)
      ZEST =ZS +C(JJ)
      IF (NOP.EQ.0 ) WRITE (6,4591) ZEST , ZBAR
 4591 FORMAT (6H ZEST=, F10.4, 6H ZBAR=, F10.4 )
      IF(ZEST.GT.ZBAR ) GO TO 3100
      GO TO 1910
C  FATHOMED DUE TO BINARY INFEASIBLE CONSTRAINT
 3100 NRED=NRED+1
      GO TO 3500
C  FATHOMED DUE TO LACK OF FREE VARIABLES
 3200 NAUG=NAUG+1
      GO TO 3500
 3300 NOPT=NOPT+1
      CALL   TIME (0,IT3)
      IF (NOP.NE.0) GO TO 3500
      ELT18 = IT3-ITO
      ELT18 =ELT18/5000.0
      WRITE (6,790) ELT18
  790 FORMAT (1H0,6H TIME=, F6.3)
      WRITE (6,3510) ZS
      WRITE (6,3600) (S(K),SB(K),K=1,L)
 3510 FORMAT (23H0 BETTER SOLUTION FOUND,5X,2HZ=,1PE15.8)
      GO TO 3500
C  BACKTRACK STEP
 3500 CONTINUE
      ICONT = ICONT +1
      NENUM=NENUM+1
      IF (NENUM.LT.KENUM) GO TO 3530
      NENUM=0
 3505 CONTINUE
      ENUM=0.0
      ENUM1 =0.0
      DO 3510 K=1,N
```

```
3509 IF (SIB(K).EQ.1) ENUM1 =ENUM1+0.5**K
     IF (SB(K).EQ.6C1B) ENUM =ENUM + .5**K
3510 CONTINUE
     CALL TIME (0,IT2)
     ELT1=IT2-IT0
     ELT2=IT2-IT1
     IT1=IT2
     ELT1=ELT1/3600.0
     ELT2=ELT2/3600.0
     IF(IT2-IT0.LT.MAXT) GO TO 3515
     MAXT=-1
     GO TO 3517
3515 CONTINUE
     IF (NOP.NE.0) GO TO 3700
3517 CONTINUE
     WRITE (6,4567) (NSG(JX),JX=1,NEL)
4567 FORMAT (15I5)
     WRITE (6,3520) ENUM,ELT1,ELT2,L
3520 FORMAT (1HO,F10.5,38H OF THE SOLUTIONS HAVE BEEN ENUMERATED,5X,
    1      15HTIME IN SECONDS,2X, 5HTOTAL, F6.3,2X,7HELAPSED F8.3,
    2         5X,2HL=,I6)
3530 CONTINUE
     IF (MAXT.LT.0) WRITE (6,3600) ((S(K),SB(K)),K=1,L)
3600 FORMAT(15(2X,I4,A1))
     IF (MAXT.LT.0) GO TO 3738
3700 NPATH=NPATH+1
3710 IF (SB(L).EQ.BLANK) GO TO 3900
     IF(SIB(L).EQ.0) GO TO 3900
     J=IABS(S(L))
     NS(J)=J
     IF (S(L).LT.0) GO TO 3735
     KK1 =(J-1)/ NEL
     KK2 = J- (KK1*NEL)
     NSG(KK2) =J
```

```
      ZS=ZS-C(J)
      DO 3725 I=1,M
 3725 BS(I)=BS(I)-A(I,J)
 3735 SB(L)=BLANK
      SIB(L) =0
      S(L)=0
      L=L-1
      IF (L.GT.0) GO TO 3710
      GO TO 3738
C  FINISHED NOW. PREPARE AND GIVE FINAL OUTPUT.
 3738 CONTINUE
      WRITE (6,3739) H1,H2
 3739 FORMAT (1H1,5X,2A8)
      DO 3740 J=1,N
 3740 S(J)=0
      DO 3742 J=1,N
      K=IABS(SMAX(J))
      IF (K.EQ.0) GO TO 3744
 3742 S(K)=1
 3744 DO 3746 K=1,N
      IF (S(K).NE.0) GO TO 3746
      SMAX(J)=-K
      J=J+1
 3746 CONTINUE
      CALL TIME (0,IT2)
      ELT1=IT2-IT0
      ELT1=ELT1/3000.0
      IF (MAXT.LT.0) GO TO 3752
      ELT40 =ELT30 + ELT1
      WRITE (6,3750) ELT1
 3750 FORMAT (30HOIMPLICIT ENUMERATION COMPLETE,5X,11HTOTAL TIME=,F8.3)
      GO TO 3758
 3752 WRITE (6,3755) ELT1
 3755 FORMAT (14HOTIME EXCEEDED,5X,11HTOTAL TIME=,F8.3)
 3758 CONTINUE
```

```
      ZBAR=0.0
      DO 3835 J=1,N
      K=IABS(SMAX(J))
      IF (SMAX(J).GT.0) ZBAR=ZBAR+C(K)
 3835 CONTINUE
      WRITE (6,3840) ZBAR
 3840 FORMAT (33HOLEAST Z BEFORE VARIABLE CHANGE=,1PE15.8)
 3800 DO 3810 K=1,N
 3810 T(K)=0
      DO 3820 K=1,N
      K1=IABS(SMAX(K))
 3820 IF (SMAX(K).GT.0) T(K1)=K1
      IF (NOP.NE.0) GO TO 3831
      WRITE (6,3830)(T(K),K=1,N)
 3830 FORMAT (15(4X,I3))
 3831 CONTINUE
      ELT3=IT3-IT0
      ELT3=ELT3/5000.0
      NITER=NPATH+NPATH-1
      WRITE(6,3850) NOPT,NRED, NAUG, NP
      WRITE (6,3851) NITER,ELT3
 3850 FORMAT (23HONO. FEASIBLE SOLUTIONS,I5/
     1         11H ZS GE ZBAR,I5,6H TIMES/
     2         24H AUGMENTATION IMPOSSIBLE,I5,6H TIMES/
     3         22H AUGMENTATION POSSIBLE,I5,6H TIMES)
 3851 FORMAT (15H NO. ITERATIONS,I5/
     1         26H LAST FEASIBLE SOLUTION AT,F8.3,9H  SECONDS)
      DO 6100 K=1,N
      IF(T(K).EQ.0) GO TO 6100
      KK1 =(K-1)/NEL +1
      KK2=K-(KK1*NEL)+NEL
      WRITE (6,4501) KK1,KK2
 4501 FORMAT (12H STATION NO.,I5,10X,12H ELEMENT NO.,I5)
 6100 CONTINUE
      WRITE (6,799) ELT30
```

```
  795 FORMAT (37H TIME FOR GENERATION OF COEFFICIENTS=, F8.3)
      WRITE (6,796) ELT17
  796 FORMAT (27H TIME FOR INITIAL SOLUTION , F8.3)
      WRITE (6,799) ELT40
  799 FORMAT (25H TOTAL TIME FOR SOLUTION , F8.3)
C  END OF FINAL OUTPUT. LOOK FOR ANOTHER PROBLEM NOW.
      READ (5,8880) ISDG
 8880 FORMAT (IF)
      GO TO 100
C  COMPLEMENT AND UNDERSCORE LAST REMAINING ENTRY IN S.
 3900 SD(L) =SCIB
      SIB(L) =1
      S(L)=-S(L)
      J=IABS(S(L))
      KK1 =(J-1)/ NEL
      KK2 = J- (KK1*NEL)
      IF (S(L).GT.0) GO TO 3950
      NSG(KK2) =0
      IF(NEL.EQ.21) IIJ1 =KK1
      IIJT = KK1 -1
      ZS=ZS-C(J)
      DO 3925 I=1,M
 3925 BS(I)=BS(I)-A(I,J)
      GO TO 1910
 3950 ZS=ZS+C(J)
      NSG(KK2) =1
      DO 3975 I=1,M
      BS(I) =BS(I) + A(I,J)
      IF(I.NE.M.AND.BS(I).LT.0.0) GO TO 8710
 3975 CONTINUE
      GO TO 1910
      END
-XQT
```

```
  20   600      1    45 ELEMENTS
  45     3     10     3  10.0
 1     9.0    272.0      0   0   0   0   0   0   0   0   0
 2     9.0    112.0      0   0   0   0   0   0   0   0   0
 3    10.0     90.0      1   0   0   0   0   0   0   0   1
 4    10.0     90.0      2   0   0   0   0   0   0   0   1
 5    17.0     80.0      3   0   0   0   0   0   0   0   1
 6    17.0     60.0      4   0   0   0   0   0   0   0   1
 7    13.0    286.0      1   0   0   0   0   0   0   0   1
 8    13.0     76.0      2   0   0   0   0   0   0   0   1
 9    20.0     68.0      5   7   0   0   0   0   0   0   2
10    26.0     58.0      6   8   0   0   0   0   0   0   2
11    10.0    324.0      9   0   0   0   0   0   0   0   0
12    11.0    325.0      0   0   0   0   0   0   0   0   0
13     6.0    314.0     11  12   0   0   0   0   0   0   2
14    22.0    203.0      7   8  13   0   0   0   0   0   3
15    11.0    234.0     18   0   0   0   0   0   0   0   1
16    12.0    184.0     13   0   0   0   0   0   0   0   1
17    12.0    135.0     14   0   0   0   0   0   0   0   1
18     3.0    145.0     15   0   0   0   0   0   0   0   1
19     7.0    142.0     16  18   0   0   0   0   0   0   2
20     4.0    101.0     19   0   0   0   0   0   0   0   1
21    50.0     97.0     20   0   0   0   0   0   0   0   1
22    14.0     92.0     21   0   0   0   0   0   0   0   1
23    27.0    115.0     15   0   0   0   0   0   0   0   1
24    29.0    121.0     15   0   0   0   0   0   0   0   1
25    26.0     86.0     14   0   0   0   0   0   0   0   1
26     6.0     60.0     17  25   0   0   0   0   0   0   2
27     9.0    117.0     17   0   0   0   0   0   0   0   1
28    24.0     78.0     22  27   0   0   0   0   0   0   2
29     4.0     47.0     14   0   0   0   0   0   0   0   1
30     5.0     40.0     14   0   0   0   0   0   0   0   1
```

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 7.0 | 50.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 32 | 4.0 | 47.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 33 | 15.0 | 53.0 | 19 | 23 | 24 | 27 | 0 | 0 | 0 | 0 | 0 | 4 |
| 34 | 3.0 | 37.0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 35 | 7.0 | 54.0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 36 | 9.0 | 63.0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 37 | 4.0 | 10.0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 38 | 7.0 | 54.0 | 28 | 26 | 36 | 34 | 0 | 0 | 0 | 0 | 0 | 4 |
| 39 | 5.0 | 48.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | 4.0 | 47.0 | 33 | 38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 41 | 21.0 | 43.0 | 9 | 10 | 40 | 31 | 32 | 29 | 30 | 39 | 0 | 6 |
| 42 | 12.0 | 22.0 | 41 | 0 | 0 | 09 | 0 | 0 | 0 | 0 | 0 | 1 |
| 43 | 6.0 | 6.0 | 37 | 0 | 0 | 08 | 0 | 0 | 0 | 0 | 0 | 1 |
| 44 | 5.0 | 5.0 | 42 | 0 | 0 | 06 | 0 | 0 | 0 | 0 | 0 | 1 |
| 45 | 5.0 | 5.0 | 42 | 0 | 0 | 08 | 0 | 0 | 0 | 0 | 0 | 1 |

43 44 45

FIN

20 600 1 45 ELEMENTS

45 8 10 3 10.0

NO. OF STATIONS 45
MAX. NO. OF STATIONS 10
CYCLE TIME 69.000

| I | TEL(I) | PWT(I) | NPR(I) | | | | | | | | NPR(NRP1) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9.0 | 272.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 9.0 | 112.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 10.0 | 90.0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 10.0 | 90.0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 17.0 | 80.0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 17.0 | 80.0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 13.0 | 236.0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 8 | 13.0 | 76.0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 20.0 | 68.0 | 5 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 10 | 20.0 | 68.0 | 6 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 11 | 10.0 | 324.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 11.0 | 323.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 6.0 | 314.0 | 11 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 14 | 22.0 | 202.0 | 7 | 8 | 13 | 0 | 0 | 0 | 0 | 0 | 3 |
| 15 | 11.0 | 254.0 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 16 | 19.0 | 184.0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 17 | 12.0 | 135.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 18 | 3.0 | 145.0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 19 | 7.0 | 142.0 | 16 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 20 | 4.0 | 101.0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 21 | 55.0 | 97.0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 22 | 14.0 | 92.0 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 23 | 27.0 | 115.0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 24 | 29.0 | 121.0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 25 | 26.0 | 86.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 26 | 6.0 | 60.0 | 17 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 27 | 5.0 | 117.0 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 28 | 24.0 | 78.0 | 22 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 29 | 4.0 | 47.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 30 | 5.0 | 48.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 31 | 7.0 | 50.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 32 | 4.0 | 47.0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 33 | 15.0 | 38.0 | 19 | 29 | 24 | 27 | 0 | 0 | 0 | 0 | 4 |
| 34 | 3.0 | 57.0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 35 | 7.0 | 54.0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 36 | 9.0 | 53.0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 37 | 4.0 | 10.0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 38 | 7.0 | 54.0 | 28 | 26 | 36 | 34 | 0 | 0 | 0 | 0 | 4 |
| 39 | 5.0 | 46.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | 4.0 | 47.0 | 35 | 38 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 41 | 21.0 | 43.0 | 9 | 10 | 40 | 31 | 32 | 29 | 30 | 39 | 8 |
| 42 | 12.0 | 22.0 | 41 | 0 | 0 | 08 | 0 | 0 | 0 | 0 | 1 |
| 43 | 6.0 | 6.0 | 37 | 0 | 0 | 08 | 0 | 0 | 0 | 0 | 1 |
| 44 | 5.0 | 5.0 | 42 | 0 | 0 | 08 | 0 | 0 | 0 | 0 | 1 |
| 45 | 5.0 | 5.0 | 42 | 0 | 0 | 08 | 0 | 0 | 0 | 0 | 1 |
| | 43 | 44 | 45 | | | | | | | | |

TOTAL NO. OF PRECEDENCE RELATIONSHIPS = 62

45 ELEMENTS

IMPLICIT ENUMERATION COMPLETE TOTAL TIME= 4.849

LEAST Z BEFORE VARIABLE CHANGE=, 0.100000000
NO. FEASIBLE SOLUTIONS 2

```
ZS GE ZBAR  979 TIMES
AUGMENTATION IMPOSSIBLE    0 TIMES
AUGMENTATION POSSIBLE 1870 TIMES
NO. ITERATIONS 1961
LAST FEASIBLE SOLUTION AT   4.661  SECONDS

STATION NO.     1          ELEMENT NO.     1
STATION NO.     1          ELEMENT NO.     2
STATION NO.     1          ELEMENT NO.     7
STATION NO.     1          ELEMENT NO.    11
STATION NO.     1          ELEMENT NO.    12
STATION NO.     1          ELEMENT NO.    13
STATION NO.     1          ELEMENT NO.    15
STATION NO.     2          ELEMENT NO.     3
STATION NO.     2          ELEMENT NO.     4
STATION NO.     2          ELEMENT NO.     5
STATION NO.     2          ELEMENT NO.     6
STATION NO.     2          ELEMENT NO.    37
STATION NO.     2          ELEMENT NO.    39
STATION NO.     2          ELEMENT NO.    43
STATION NO.     3          ELEMENT NO.     8
STATION NO.     3          ELEMENT NO.     9
STATION NO.     3          ELEMENT NO.    14
STATION NO.     3          ELEMENT NO.    18
STATION NO.     3          ELEMENT NO.    29
STATION NO.     3          ELEMENT NO.    31
STATION NO.     4          ELEMENT NO.    16
STATION NO.     4          ELEMENT NO.    17
STATION NO.     4          ELEMENT NO.    19
STATION NO.     4          ELEMENT NO.    20
STATION NO.     4          ELEMENT NO.    23
STATION NO.     5          ELEMENT NO.    10
STATION NO.     5          ELEMENT NO.    24
STATION NO.     5          ELEMENT NO.    27
STATION NO.     5          ELEMENT NO.    33
```

```
STATION NO.     6          ELEMENT NO.    21
STATION NO.     6          ELEMENT NO.    22
STATION NO.     7          ELEMENT NO.    25
STATION NO.     7          ELEMENT NO.    26
STATION NO.     7          ELEMENT NO.    28
STATION NO.     7          ELEMENT NO.    32
STATION NO.     7          ELEMENT NO.    36
STATION NO.     8          ELEMENT NO.    30
STATION NO.     6          ELEMENT NO.    34
STATION NO.     8          ELEMENT NO.    35
STATION NO.     8          ELEMENT NO.    38
STATION NO.     8          ELEMENT NO.    40
STATION NO.     8          ELEMENT NO.    41
STATION NO.     8          ELEMENT NO.    42
STATION NO.     8          ELEMENT NO.    44
STATION NO.     8          ELEMENT NO.    45

TIME FOR GENERATION OF COEFFICIENTS=     .100

TIME FOR INITIAL SOLUTION      .048

TOTAL TIME FOR SOLUTION     4.949
```

# BIBLIOGRAPHY


1. Arcus, A. L., "COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines," in Buffa, E. S., editor, Readings in Production and Operations Management, John Wiley and Sons, New York, 1966.
2. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero One Variables," Operations Research, 13, 517-546 (1965).
3. ————, "Discrete Programming by the Filter Method," Operations Research, 15, 915-957 (1967).
4. Balinski, M. L., "Integer Programming: Methods, Uses and Computation," Management Science, 12, 253-313 (1965).
5. Bowman, E. H., "Assembly Line Balancing by Linear Programming," Operations Research, 8, 385-389 (1960).
6. Buffa, E. S., Production - Inventory Systems, Richard D. Irwin. Inc., Illinois, 229-67 (1968).
7. Cauley, J. M., "A Review of Assembly Line Balancing Algorithms," Proceedings of 19th Annual Conference, American Institute of Industrial Engineers, New York, (1968).
8. Geoffrion, A. M., and Nelson, A. B., Users Instructions for 0-1 Integer Linear Programming Code RIP30C, RM 5627-PR, The Rand Corporation, Santa Monica, California (1968).
9. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas' Method," SIAM Review, 9, 178-190 (1967).
10. Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, 17, 437-454 (1969).

11. Glover, F., "Surrogate Constraints," Operations Research, 16, 741-749 (1968).

12. Gue, R. L., Ligget, J. C., and Cain, K. C., "Analysis of Algorithms for the Zero-One Programming Problem," Communications of the ACM, 11, 837-844 (1968).

13. Gutjahr, A. L., and Nemhauser, G. L., "An Algorithm for the Line Balancing Problem," Management Science, 11, 308-315 (1964).

14. Held, M., Karp, R. M., and Shareshian, R., "Assembly Line Balancing-Dynamic Programming with Precedence Constraints," Operations Research, 11, 442-459 (1963).

15. Helgeson, W. P., and Birnie, D. D., "Assembly Line Balancing Using the Ranked Positional Weight Technique," Journal of Industrial Engineering, 12, 394-398 (1961).

16. Heskia Heskiaoff, "An Heuristic Method for Balancing Assembly Lines," Western Electric Engineer, 9-17, Oct. (1968).

17. Hoffman, T. R., "Assembly Line Balancing with a Precedence Matrix," Management Science, 9, 551-563 (1963).

18. Ignall, E., "A Review of Assembly Line Balancing," Journal of Ind. Engg., 16, 244-254 (1965).

19. Jackson, J. R., "A Computing Procedure for a Line Balancing Problem," Management Science, 9, 551-563 (1963).

20. Jaeschke, G., "Eine allgemaine Methode Zur Lösung kombinatorisher Probleme," Ablauf-und Planung forshung 5, 133-153 (1964).

21. Kilbridge, M. D., and Wester, L., "An Heuristic Method of Assembly Line Balancing," Journal of Ind. Engg., 12, 292-298 (1961).

22. Klein, M., "On Assembly Line Balancing," Operations Research 11, 274-281 (1963).

23. Lawler, E. L., and Wood, D. E., "Branch and Bound Methods: A Servey," Operations Research 14, 699-719 (1966).

24. Mansoor, E. M., "Assembly Line Balancing - A Heuristic Algorithm for Variable Operator Performance Levels," Journal of Ind. Engg., 19, 618-629 (1968).

25. Mastor, A. A., "An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques," Unpublished Ph.D. Dissertation, UCLA, 1966. Also reported in (6).

26. Mertens, P., "Assembly line Balancing by Partial Enumeration," Ablauf und Planung forshung 8, 429-433 (1967).

27. Moodie, C. L., and Mandeville, D. E., "Project Resource Balancing by Assembly Line Balancing Techniques," Journal of Ind. Engg. 17, 377-383 (1966).

28. Moodie, C. L., and Young., H. H., "A Heuristic Method of Assembly Line Balancing for Assumptions of Constant or Variable Work Element Times," Journal of Ind. Engg. 16, 23-29 (1965).

29. Pierce, R. D., "Advanced Assembly Methods Program - Update on IITRI," Assembly Engg. 4, 18-24, April (1968).

30. Reiter, "On Assembly Line Balancing Problems," Operations Research, 17, 685-700 (1969).

31. Salveson, M. E., "The Assembly Line Balancing Problem," Journal of Ind. Engg. 6, 18-25 (1955).

32. Tonge, F. M., "Summary of the Heuristic Line Balancing Procedure," Management Science 7, 21-42 (1960).

33. Wester, L., and Kilbridge, M. D., "A Review of Analytical Systems of Line Balancing," Operations Research 10, 626-638 (1962).

34. _____, "The Assembly Line Model - Mix Sequencing Problem," The Proceedings of the Third International Conference on Operations Research 1963, Dunod, Paris, 1964.

35. White, W. W., Comments on a Paper by Bowman - Operations Research 9, 274-276 (1961).