An integer programming approach to mixed model assembly line balancing.

Mehmet Melih Akgun

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/1096

This online database contains the full-text of PhD dissertations and Masters’ theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not-filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S’il manque des pages, veuillez communiquer avec l’université qui a conféré le grade.

La qualité d’impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l’aide d’un ruban usé ou si l’université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l’objet d’un droit d’auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d’auteur. SRC 1970, c. C-30. Veuillez prendre connaissance des formules d’autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L’AVONS RÉCU
AN INTEGER PROGRAMMING APPROACH
TO MIXED MODEL ASSEMBLY LINE BALANCING

by

MEHMET MELIH AKGUN

'A Thesis Submitted to the Faculty of Graduate Studies through the Department of Industrial Engineering in Partial Fulfillment of the requirements for the Degree of Master of Applied Science at The University of Windsor

Windsor, Ontario, Canada
1981
ABSTRACT

A multiproduct assembly line balancing problem is investigated in this study. An integer programming model is suggested to handle the grouping requirements of similar tasks of different products to the same stations.

A general formulation of the problem is given and two approaches are presented for handling the grouping requirements. In the first approach the grouping requirements are met without creating additional stations. In the second approach, however, the cost of splitting similar tasks and the cost of creating new stations which may be required to meet these requirements are taken into consideration.
ACKNOWLEDGEMENTS

The author wishes to express his sincerest gratitude to Dr. A. Raouf for his valuable supervision and encouragements, and to Dr. R.S. Lashkari for his interest and helps throughout this study.

The author is also grateful to Dr. M. Shridhar for his valuable suggestions.

My sincere thanks due to Dr. Yuceer for his helps and his patience in reading the drafts.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td></td>
<td>ACKNOWLEDGEMENT</td>
<td>iif</td>
</tr>
<tr>
<td></td>
<td>TABLE OF CONTENTS</td>
<td>iv</td>
</tr>
<tr>
<td></td>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>LITERATURE SURVEY</td>
<td>7</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>A GENERAL 0-1 INTEGER PROGRAMMING MODEL OF MIXED MODEL ASSEMBLY LINE BALANCING PROBLEM</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.1 Mathematical formulation</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.2 Solution method</td>
<td>25</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>GROUPING VERSUS NUMBER OF STATIONS</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>4.1 Problem definition</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>4.2 Mathematical formulation</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>4.3 Solution procedure</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>4.4 Operational results</td>
<td>45</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>COST ASPECTS OF THE PROBLEM</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>5.1 Definition of costs</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>5.2 Mathematical formulation</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>5.3 Solution procedure</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>5.4 Operational results</td>
<td>62</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>CONCLUSIONS</td>
<td>70</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Proof</td>
<td>72</td>
</tr>
</tbody>
</table>
APPENDIX B  Flow chart  
APPENDIX C1  Computer program 1  
APPENDIX C2  Computer program 2  
APPENDIX D  Sample problems  
REFERENCES  
VITA AUCTORIS
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Precedence diagram of the example problem</td>
<td>4</td>
</tr>
<tr>
<td>4.1</td>
<td>Precedence diagram of the sample problem</td>
<td>41</td>
</tr>
<tr>
<td>D.1</td>
<td>Precedence diagram of test problem 1</td>
<td>106</td>
</tr>
<tr>
<td>D.2</td>
<td>Precedence diagram of test problem 2</td>
<td>107</td>
</tr>
<tr>
<td>D.3</td>
<td>Precedence diagram of test problem 3</td>
<td>108</td>
</tr>
<tr>
<td>D.4</td>
<td>Precedence diagram of test problem 4</td>
<td>109</td>
</tr>
<tr>
<td>D.5</td>
<td>Precedence diagram of test problem 5</td>
<td>110</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Solution of test problem 1</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>Solution of test problem 2</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Solution of test problem 3</td>
<td>49</td>
</tr>
<tr>
<td>4.4</td>
<td>Solution of test problem 4</td>
<td>50</td>
</tr>
<tr>
<td>4.5</td>
<td>Solution of test problem 5</td>
<td>51</td>
</tr>
<tr>
<td>5.1</td>
<td>Cost incorporated solution of test problem 1</td>
<td>64</td>
</tr>
<tr>
<td>5.2</td>
<td>Cost incorporated solution of test problem 2</td>
<td>65</td>
</tr>
<tr>
<td>5.3</td>
<td>Cost incorporated solution of test problem 3</td>
<td>66</td>
</tr>
<tr>
<td>5.4</td>
<td>Cost incorporated solution of test problem 4</td>
<td>67</td>
</tr>
<tr>
<td>5.5</td>
<td>Cost incorporated solution of test problem 5</td>
<td>68</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

The assembly line can be defined as the movement of the workpiece from one task to the next. The tasks that must be performed on the product are divided among workers (or worker groups) so that a given worker performs the same operation on the product which passes before him. The balance of an assembly line is the assignment of these tasks along a production line in order to increase the production efficiency.

Restrictions of Assembly Line Balancing (ALB):

Each task requires a known operation time per unit, independent of when and where it is performed.

The tasks will be assigned to different operation regions called stations which are arranged sequentially. A workpiece can move to the next station after the task assigned to the present station is completed. The total time of the tasks assigned to any station can not exceed a certain amount of time called cycle time.

Another requirement of the problem is the precedence relations. These relations restrict the order in which the tasks can be performed. They must be assigned to stations in such a way that their predetermined sequences are met.
The tasks cannot be split among stations; that is to say, each task is assigned exactly once and performed at a single station only.

**Single Item ALB:**

Single item ALB is the balance of a line, which pertains to a single product type. More explicitly, there is only one type of product in production, and the line is balanced to consider only this product and the tasks which belong to it. The maximum work load of stations are the cycle time of this item.

**Mixed Model ALB:**

The mixed model ALB (MALS) can be interpreted as an extension of single item ALB. More than one product may be assembled on the same assembly line. Instead of assembling each product at a time and making batches intermittently for inventory, it may be required to keep every product on production and have a steady flow of different product types.

These types of assembly line balancing problems undoubtedly represent more complex design and operating problems. As far as the following points are concerned, they are quite different than the single model ALB.

The cycle time of every product can be different. Every product type may have a different job flow pattern; in other
words, different products may use the same tasks but with a different order.

The tasks which are used by different product types may be different. It is always possible to find some tasks which are used by some product types but not by other product types.

The following illustration is given in order to clarify the MALB problem:

Figure(1.1) represents the flows of two products separately. The solid lines represent the pattern of the first product. The dotted lines represent the pattern of the second product. The circled numbers are task identifiers. The uncircled numbers (north-east of task numbers) are the required times of the tasks. 1, 2, 3, 4, 5, 6 are the tasks of the first product. 1', 2', 3', 4', 5', 6' are the tasks of the second product. 1 and 1', 2 and 2', 3 and 3', 5 and 5', 6 and 6' are similar tasks which may be required to be assigned to the same station.

Criteria of Production Efficiency and Objectives of MALB: 

In most of the MALB problems, production efficiency is increased using the following criteria:

i) minimum cycle time

ii) minimum delay time

iii) minimum station number

The first criterion can be interpreted as minimizing the time spent by a workpiece at each work station. Although it may
Precedence diagram of the example problem

Figure (1.1)
be considered a good criterion, a number of drawbacks are associated with it. The major drawback is that forcing the production for a very low cycle time may be very expensive. This can be performed with some overtime works, second shifts, duplicating the equipment, and so forth. In order to compensate for this cost, demand must be very high, which may not always be the case. Another inconvenience of this criterion is the computational difficulties. Finding a minimum cycle time in a given number of stations have been done by trial and error rather than mathematically.

The remaining two criteria can be defined respectively as the minimization of station numbers and the balance delay which can be defined as the quotient of total idle time to total work time in a given cycle time C. Although they seem to be two different criteria, the minimization of the number of stations minimizes the total balance delay as well. If N represents the number of stations on the line, and Ti is the amount of time assigned to the ith station, for a given cycle time C, the balance delay can be written as:

\[ \text{delay} = \frac{\sum_{i=1}^{N} (C-T_i)}{\sum_{i=1}^{N} T_i} \]

\[ \text{minimum delay} = \min[N - \frac{\sum_{i=1}^{N} T_i}{\sum_{i=1}^{N} T_i}] \]
Since $T_i$ and $C$ are constants the expression becomes:

$$\min \left( \frac{NC}{\text{constant}-1} \right) = \min(NC) = C \min(N)$$

As can be seen, the minimization of the total delay minimizes the number of stations as well. In this research, the above expression forms the objective function.

The background of the work is presented in Chapter 2. Chapter 3 gives a general 0-1 integer programming formulation of MALB. In Chapter 4, the grouping requirements are dealt with. A new approach has been developed for the grouping requirements of MALB. The problem of assigning similar tasks of different products to the same station has been handled with a different approach. Rather than treating these requirements as constraints of the problem, they are relaxed and forced by the objective function to be assigned to the same station if an extra station is not needed in order to meet these requirements. Chapter 5 deals with the cost aspects of the problem. In this chapter the cost of the duplication of a task for different products and the cost of opening a new station are considered and the objective function presented in Chapter 4 is modified to incorporate these costs.
CHAPTER 2

LITERATURE SURVEY

Numerous methods have been proposed to establish the assembly line balancing in various research articles. The problem of assigning tasks to stations is of combinatorial type and the solution procedures can be classified as branch-and-bound methods and heuristic approaches. Branch-and-bound methods can be characterized by branching from a current partial solution to another if it yields a feasible solution, or fathoming if it does not. The solutions generated by these procedures are optimal, but they were not considered to be practical because of the computational requirements. On the other hand some heuristic techniques have been proposed. Since they seem more practical from the computational viewpoint, most researchers have preferred to work with these techniques. However, they do not guarantee optimal results and can only produce near optimal solutions.

The first published article (1955) on assembly line balancing belongs to Salveson[24]. He proposed two methods in order to handle this problem. He suggested one method for the lines with few precedence constraints and another method for the lines with considerably more precedence constraints. The main formal solution method Salveson proposes is a linear programming model which allows tasks to be split among stations.
Jackson[11], in 1956 presented a method which minimizes the number of stations for a given cycle time. The method starts with the construction of all feasible first work stations; then, for all task combinations of first work stations, it finds alternatives for second work stations, and so forth. A minimum number of stations is then obtained for a selected cycle time. His method is feasible for hand calculations of relatively small assembly line problems, but it requires a great amount of storage and running time when a moderately large size problem is solved on a computer.

Tonge[28] imbedded Jackson's method in his procedure, and developed another heuristic procedure. His method consists of three phases:

Phase 1: Repeated simplification of the initial problem by grouping adjacent work elements into compound tasks.

Phase 2: Assignment of these compound tasks to stations.

Phase 3: Rearranging the tasks among stations to reduce the cycle time.

Bowman[2] presented two integer-linear programming models, one of which was improved by White[29]. Bowman's both formulations require a large number of variables and constraints. His first model requires about 135 constraints and 56 variables for a problem of 9 tasks. An improvement of his first formulation was accomplished by White. In his published
comment, White reformulated Bowman's first model and introduced the concept of 0-1 programming to this specific problem. From this point of view, his improvement is extremely valuable. He could also represent the same problem with 71 constraints and 56 variables rather than 135 and 56, respectively.

In the heuristic technique of Kilbridge and Wester[12], column labels are added to the precedence diagram. In the first column, all tasks having no predecessors are entered; in column \( K \), all tasks whose predecessors labeled by columns \( K-1, \ldots \) \( 1 \) are entered. In each column, the elements have two properties: permutability within columns and transferability to adjacent columns. The remaining part of the algorithm deals with searching for the optimal solution by a systematic trial and error for a given cycle time under these column properties. Another paper[13] presented by the same authors deals with the model-mix sequencing problem. They propose two analytical systems in order to determine the best sequence in the flow of models when a variety of models of the same general product are intermixed on one assembly line. One review paper[14] also has been written by the same authors.

Holgeson and Birnie[7] proposed the "Ranked Positional Weight" technique (RPW). The weight of each task is defined as the sum of its time and the times of its successors. Then, these tasks are sorted with respect to their weight in descending order. One starts to assign these tasks to stations using this sorted sequence. If a task violates the precedence
relation or takes longer than the time remaining in the current station, it is skipped. This process continues until no further tasks can be assigned to the station. At that point the assignment of the next station starts off with the first skipped task. This technique seems quite simple and very convenient for manual calculations, although it is not guaranteed for a minimum station balance.

Arcus[1], in his heuristic technique which randomly generates a large number of feasible sequences and chooses the one that gives the fewest stations, presented a methodology in order to extend his algorithm for a mixed model case. His extension is mainly the multiplication of the standard times of tasks by the relative frequencies of the related products and then using these average times for his single model line balancing algorithm.

The Held, Karp, and Sahasranam[6] algorithm uses a dynamic programming approach to determine which feasible sequence to be used for the assignment of work elements to the stations. For small problems their computational procedure first finds all feasible subsets of a partially ordered set and computes, recursively, the cost of each feasible subset. After this has been completed, the optimal feasible sequence is determined recursively. For larger problems, their technique also includes a method of successive approximation that uses several judgemental decisions made by the authors.

In 1963, Hoffman[9] proposed a matrix manipulation
technique. His method begins with a precedence matrix, whose elements give the same information as the precedence diagram. If the elements of row i immediately precede the element of column j, a '1' is placed in position (i,j). All other entries are zero. In his paper using this precedence matrix, he gives an algorithm and its Fortran coding.

The research which has been done by Hanks and [16] can be interpreted as a refinement of the ranked positional weight (RPW) of Helgeson and Birnie[7]. His key idea is applying RPW but, at the same time, keeping track of the total idle time. If the total idle time exceeds a predetermined undesirable value, he suggests to backtrack and remove from the current station some of the tasks which have already been assigned, and then try RPW from this point. His procedure guarantees an optimal solution, but the amount of work may be quite large even for moderate size problems.

Mooijie and Young[20] have developed a two phase heuristic procedure for balancing lines. In the first phase, a preliminary balance is obtained using the "largest candidate rule". This phase consists of constructing work stations sequentially in such a way that at each stage one selects the task which has the largest time among the feasible tasks and assigns at the current station. In the second phase, as in Tonge's phase 3, heuristics are used to shift tasks between stations in order to minimize the total idle time and to smoothen the stations. Their procedure also allows task
performance times to be variable.

Two exhaustive reviews have been published by Ignall[10] and Cauley[3] in 1965 and 1968 respectively.

Lehman presented another heuristic approach for assigning models to assembly lines[15]. His method is the assignment of these models to the assembly lines, considering the quantity to be produced and the work load of every model. Work capacities of lines are also bounded by lower and upper bounds. His method can be interpreted as an assignment problem of models to assembly lines considering the required work and the capacities of facilities, rather than an assembly line balancing.

In 1971 Thangavelu and Shetty[25] presented an improved version of the Bowman-White 0-1 integer programming formulation of the assembly line balancing problem. In order to solve the problem they simplified and eliminated some steps of Geoffrion's[5] 0-1 integer programming algorithm. As far as the number of constraints is concerned, the model which has been developed by Thangavelu and Shetty was the best. They could define the problem with only \((M+N+R+1)\) inequality constraints where:

\[ M = \text{total number of stations under consideration} \]

where: \(N = \text{total number of tasks}\)

\(R = \text{total number of precedence relations}\).

In the model developed by Patterson and Albracht[22], the number of variables required to define the problem is
decreased approximately by 50%. This was due to a variable elimination criterion which was introduced by them. The criterion is quite simple but also quite useful in the elimination of the variables which correspond to physically impossible assignments. They defined an earliest (Ei) and latest (Li) assignable station for every task ‘i’ using the following equations:

\[
E_i = \begin{cases} 
1 & \text{for } (T_i + \sum_{j \in P_i} T_j) = 0 \\
\frac{(T_i + \sum_{j \in P_i} T_j) + C}{C} & \text{otherwise} 
\end{cases} 
\]

\[
L_i = \begin{cases} 
M & \text{for } (T_i + \sum_{j \in S_i} T_j) = 0 \\
M + 1 - \frac{(T_i + \sum_{j \in S_i} T_j) + C}{C} & \text{otherwise} 
\end{cases} 
\]

- \(N\) = total number of tasks
- \(C\) = cycle time
- \(M\) = total number of stations
- \(P_i\) = set of tasks which precede task \(i\)
- \(S_i\) = set of tasks which succeed task \(i\)
- \(T_j\) = time required to perform task \(j\).

In 1970, Roberts and Villa[23] presented a 0-1 integer programming formulation for a mixed product ALB problem. Their
formulation is of more theoretical than practical interest. The number of constraints and variables are quite large, and make the problem a difficult undertaking even for modest size problems.

Thomopoulos presented another balancing and sequencing technique for mixed model lines[26]. He described a procedure for adapting the single model line balancing technique developed by Kilbridge and Wester[12] to mixed model schedules. He also introduced a sequencing procedure for determining the order in which models are to flow down the line. In his second article[27], Thomopoulos' approach to mixed model cases was conceptually very similar to Arcus' research. The total time required of a task is found by multiplying the number of units of every product by the corresponding work time and taking the sum. He also preferred to use lower and upper bounds for the work load of stations rather than a predetermined fixed cycle time.

Other research which has been carried out on mixed-model assembly line balancing belongs to Macaskill[16]. The main concept of his work is superimposing different job flow patterns of different products and balancing the combined job flow diagram by grouping similar tasks of different products and assigning the whole group to the same station. He also developed a simulation approach to PALB[17] and gave a sequencing procedure.

On establishing the production schedule for the
mixed-model assembly line production systems, Hiraki[8] presented a branch and bound algorithm.

Another study which pertains to the sequencing of models in mixed model lines belongs to Dar-El and Cother[4]. They developed an algorithm to maximize the operator efficiency of an assembly line of fixed length for a given production requirement.

In a recent paper, Okamura and Yamashina[21] developed another formulation of the mixed model sequencing model in order to minimize the risk of stopping the conveyor under the circumstances of system variability and, secondly, to develop an efficient heuristic method for large scale, mixed-model assembly lines. Their method assumes that a suitable mixed-model balance has already been achieved and that the belt conveyor speed and length of every station have been determined.
CHAPTER 3

A GENERAL 0-1 INTEGER PROGRAMMING MODEL
OF MIXED MODEL ASSEMBLY LINE BALANCING

3.1 MATHEMATICAL FORMULATION:

The mathematical model developed in this chapter is a
general purpose formulation for the balancing of a mixed-model
assembly line. The proposed model of this research reduces the
number of constraints as compared to other models in the
literature. The coefficients introduced in the constraints
(3.4) and (3.5) as well as the new formulation of grouping
requirements (3.12), (3.13) considerably reduce the number of
equations.

The variables and the parameters of the model are listed
below:

\[ X(i,j,k) = \begin{cases} 
1 & \text{if task } i \text{ of product } k \text{ is assigned to station } j \\
0 & \text{otherwise} 
\end{cases} \]

\[ T \quad \text{time period considered by the management as one shift} \]

\[ N \quad \text{total number of product types} \]

\[ M \quad \text{total number of stations under consideration} \]

\[ SJ(k) \quad \text{set of tasks required for product } k \]

\[ J(k) \quad \text{total number of required tasks for product } k \]
\[ J = \sum_{k=1}^{N} J(k) \]  

- total number of tasks for all products

\[ S(i,k) \]  

- subset of all tasks that must succeed task \( i \) of product \( k \)

\[ P(i,k) \]  

- subset of all tasks that must precede task \( i \) of product \( k \)

\[ E(i,k) \]  

- earliest station where task \( i \) of product \( k \) can be assigned

\[ L(i,k) \]  

- latest station where task \( i \) of product \( k \) can be assigned

\[ N(i) \]  

- number of products which need task \( i \)

\[ n(k) \]  

- number of product \( k \) to be produced within a shift time \( T \)

\[ C(i,j,k) \]  

- cost of assigning task \( i \) of product \( k \) to station \( j \)

\[ t(i,k) \]  

- time required to complete task \( i \) of product \( k \)

**Variable Elimination:**

Although every possible assignment of each task of each product requires one variable \( X(i,j,k) \), precedence relations may allow the elimination of some of the variables. Obviously if the total number of stations is greater than one, it is unnecessary to consider the assignment of the first task to the last station. Or if the \( i \)th task is under consideration and the total time required to perform all tasks which precede task \( i \) is greater than the shift time, it is impossible to assign task \( i \) to the first station, and thus the corresponding decision
variables $X(i,j,k)$ will be eliminated. This idea was first suggested by Patterson and Albracht[22] for single item ALB. Their approach to the mixed model case can be expressed as follows:

$$E(i,k) = \begin{cases} 1 & \text{if } n(k)t(j,k) + \sum_{j \in P(i,k)} n(k)t(j,k) = 0 \\ \frac{(n(k)t(i,k) + \sum_{j \in S(i,k)} n(k)t(j,k))}{T} + & \text{otherwise} \end{cases} \quad (3.1.a)$$

$$L(i,k) = \begin{cases} M & \text{if } n(k)t(i,k) + \sum_{j \in S(i,k)} n(k)t(j,k) = 0 \\ M+1-\frac{(n(k)t(i,k) + \sum_{j \in S(i,k)} n(k)t(j,k))}{T} + & \text{otherwise} \end{cases} \quad (3.2.a)$$

where the notation $[n]$ denotes the smallest integer greater or equal to $n$. Equation (3.1.b) indicates that the latest station of task $i$ of product $k$ is calculated by dividing the total required work time of this task and all its predecessors to the shift time. Equation (3.2.b) indicates that the latest station of a task can be found by calculating the minimum number of stations necessary for this particular task and all its successors. This number is the division of the total work time of these tasks to the shift time. Finally the latest station is
found by subtracting this figure from \( N+1 \). Equations (3.1.a) and (3.2.a) are for the dummy tasks of work time zero which may be necessary as initial or final tasks.

**Occurrence constraints:**

The constraints which assure that all tasks of all products are performed and each task is assigned at only one station can be expressed as:

\[
\sum_{j=E(i,k)} X(i,j,k) = 1 \quad \text{for} \quad k=1,2,\ldots,N \quad \text{and} \quad i=1,2,\ldots,J(k)
\]

(3.3)

Since the solution procedure requires the constraints to be in inequality form, instead of replacing the above \( N \) equalities with \( 2J \) inequalities, the following constraints are used:

\[
\sum_{k=1}^{N} d(k) \sum_{i \in SJ(k)} \sum_{j=E(i,k)} X(i,j,k) \geq \sum_{k=1}^{N} d(k)J(k)
\]

(3.4)

Where:

\[
d(1) = 1
\]

\[
d(k) = d(k-1) \sum_{s=1}^{k-1} [L(i,s) - E(i,s) + 1] + 1 \quad k=2,3,\ldots,N
\]

\( d(k) \) is the coefficient pertaining to variables of product \( k \) and \( d(1) \) is always greater than the number of variables pertaining to
product \( k-1 \) multiplied by their coefficient \( (k-1) \).

The above constraint implies that among the variables corresponding to product \( k \) at least \( J(k) \) variables will be augmented to 1. Even though all variables corresponding to products \( 1, 2, \ldots, k-1 \) are incremented to 1, due to the coefficients \( \beta(k) \) the constraint will not be satisfied unless \( J(k) \) variables of product \( k \) are augmented to 1.

Another set of restrictions which must be accompanied with constraint (3.4) are:

\[
\sum_{k=1}^{N(i)} \beta(k) \sum_{j=E(i,k)}^{N(i)} x(i,j,k) \leq \sum_{k=1}^{\max\{J(1), J(2), \ldots, J(N)\}} \beta(k) \quad i=1, 2, \ldots, \max\{J(1), J(2), \ldots, J(N)\}
\]

(3.5)

where:

\[
\beta(1) = 1
\]

\[
\beta(k) = [L(i, k-1) - E(i, k-1) + 1] (k-1) + 1 \quad k=2, 3, \ldots, N
\]

\( \beta(k) \) is the coefficient corresponding to product \( k \) and is always greater than the number of assignable stations of product \( k-1 \) times their coefficient \( \beta(k-1) \).

This set of constraints ensure that among the variables corresponding to task \( i \) at most \( N(i) \) variables will be augmented to 1. Furthermore, due to coefficients each of these variables must correspond to different products. Considering the constraint (3.4), \( J(k) \) variables have been forced to be assigned 1 for every product \( k \). This constraint restricts that these \( J(k) \) variables can not have the same task subscripts; that
is, a particular task of a particular product must be assigned to only one station.

Using (3.4) and (3.5) the occurrence constraints are expressed with only \( \max\{J(1), J(2), \ldots, J(N)\} + 1 \) equations rather than \( 2J \).

Precedence constraints:

For each precedence relation where task \( p \) of product \( k \) must immediately precede task \( q \) of the same product, the following constraints are required:

\[
\sum_{i=E(p,k)} iX(p,i,k) \leq \sum_{j=E(q,k)} jX(q,j,k) \quad k=1,2,\ldots,N \quad (p,q) \in R(k)
\]

where:

\[
R(k) = \{ (p,q) \mid \text{task } p \text{ of product } k \text{ immediately precedes task } q \text{ of the same product} \}
\]

If the cardinality of \( R(k) \) is \( r(k) \), the number of precedence constraints would be \( \sum_{k=1}^{N} r(k) \)

Shift time constraints:

The requirement that the total time of tasks assigned to a station can not exceed the shift time is expressed by:

\[
\sum_{k=1}^{N} \sum_{i=1}^{J(k)} n(k)X(i,j,k)t(i,k) \leq T \quad j=1,2,\ldots,N \quad (3.7)
\]
Zoning constraints:

Although the constraints (3.4), (3.5), (3.6), (3.7) are sufficient to define the problem, this mathematical model can be extended for some other requirements. If two tasks p and q are given and it is required not to schedule them at the same station, this can be expressed by:

If p succeeds q,

\[ \sum_{j \in E(p,k)} jX(p,j,k) \geq 1 + \sum_{j \in E(q,k)} jX(q,j,k) \quad k=1,2,\ldots,N \quad (3.8) \]

Another requirement may be the performing of some similar works of different products at the same station. To achieve this let \( G(t) = \{(i,k) | \text{task } i \text{ of product } k \text{ belongs to the } t \text{th grouping and is required to be assigned to the same station} \} \) and \( g(t) = \text{cardinality of } G(t) \).

For example if there is a grouping requirement for two tasks \( (G(t) = \{(i,k) : (j,1) \} \) and \( g(t) = 2 \)), this can be expressed with the following equalities as suggested by Roberts and Villa[23]:

\[ X(i,n,k) = X(j,n,1) \quad n=1,2,\ldots,M \quad (3.9) \]

Writing them in inequality form:

\[ X(i,n,k) \geq X(j,n,1) \quad n=1,2,\ldots,M \quad (3.10) \]

\[ X(i,n,k) \leq X(j,n,1) \quad n=1,2,\ldots,M \quad (3.11) \]

Therefore \( 2M[g(t)-1] \) inequalities are required for a particular
station \( t \). In this formulation this requirement is expressed in a different way as:

\[
\sum_{(i,k) \in G(t)} X(i,j,k) \sum_{j=1}^{M} g(t) y(j) \quad j=1,2,\ldots,M
\]

(3.12)

\[
\sum_{j=1}^{N} \mathcal{f}(j) \geq 1 \quad \text{where } y(j) = (0,1)
\]

(3.13)

Equation (3.12) guarantees that the sum of all variables corresponding to set \( G(t) \) will be either 0 or \( g(t) \). Equation (3.13) ensures that this sum must be equal to \( g(t) \) at least once: in other words, at least for one of the stations. Although these two restrictions may seem to allow more than one \( y(j) \) to be augmented to 1, the equation (3.5) prevents this incidence. In particular, the assignment of a task \( i \) of product \( k \) to two or more stations has been prevented by equation (3.5).

This formulation reveals that \( M+1 \) constraints and \( M \) extra variables are sufficient to satisfy the requirements instead of \( 2M[(g(t)-1)] \) constraints. The important aspect of formulating the problem in this fashion is that the additional number of constraints and variables are independent of the numbers of elements in sets \( G \). Evidently the result is a more efficient formulation than the conventional approaches (equations 3.9, 3.10, 3.11).

\textbf{Objective function:}

Given the work times of all the tasks of all the
products and the required shift time, the absolute minimum number of stations \( N_0 \) can be calculated by the following expression:

\[
N_0 = \left[ \frac{\sum_{k=1}^{N} \sum_{i=1}^{J(k)} t(i,k)}{T} \right]^+ \tag{3.14}
\]

The objective function of the model is basically Bowman's[2] cost explosion function, which, in this formulation, has been modified to incorporate the product types for mixed model assembly line balancing.

\[
\text{Min} \sum_{k=1}^{J(k)} \sum_{i=1}^{N} \sum_{j \in \mathcal{E}(j,k)} C(i,j,k) X(i,j,k) \tag{3.15}
\]

where:

\[
C(i,j,k) = \begin{cases} 
\sum_{i \in \mathcal{F}} n(k)t(i,k) [\sum_{i \in \mathcal{F}} n(k)t(i,k) + 1] & \text{for } j = 1, 2, \ldots, N_0-1, \\
0 & \text{for } j \in \mathcal{N}_0+1, \ldots, N \\
\text{otherwise} & 
\end{cases}
\]

\( \mathcal{F} = \{ i \mid \text{task } i \text{ does not precede any other element} \} \).  

This objective function makes later stations very costly in order to force the tasks to be assigned to the earlier stations. Since stations 1, 2, \ldots, \( N_0 \) must certainly be used, there is no cost assignment for them. Only final tasks (tasks without successors) have positive costs.

The nature of this cost explosion on the final tasks is
simply to make one unit of a later assignment more costly than the sum of all the preceding station assignments.

3.2 SOLUTION METHOD

This mathematical model is a 0-1 integer programming problem. Therefore, it can be handled by any 0-1 integer-linear programming method. In particular, some pilot studies have been performed using Geoffrion's[5] algorithm without making any modification in the procedure. During this study it has been observed that a general 0-1 integer-linear programming algorithm for this problem requires large amounts of execution time and core space. Ultimately, on the basis of Thangavelu and Shetty's paper, Geoffrion's algorithm is considerably modified in order to increase its efficiency when applied to this particular model. These modifications are explained in Chapter 4.
CHAPTER 4

GROUPING VERSUS NUMBER OF STATIONS

4.1 PROBLEM DEFINITION:

The main idea behind this formulation is the grouping of similar tasks of different products and the forcing of the tasks of the same group to the same station. If a particular task is used for different products and if the repetition of this task for every product is considered as a different task, it is always preferred to assign them to the same station provided the station time does not exceed the shift time. This requirement has been handled by identifying these similar tasks and assigning the whole similar task group to a station. In some cases the assignment of these tasks to the same station may result in more stations than if grouping requirements were eliminated. For instance if the problem in Figure(1.1) is balanced for a shift time of 30, at least 3 stations will suffice to meet the grouping requirements, and one possible solution may be:

Station 1 ---> 1, 1', 3, 3'
Station 2 ---> 2, 2', 5, 5', 4
Station 3 ---> 6, 6'

Although the stations have large idle times (7, 4, 24), and the line has very low efficiency (0.61), 3 station balance is inevitable.

Further inspection of figure(1.1) may yield a better
solution, for the same line consisting of 2 stations. The only sacrifice would then be the grouping requirement of tasks 2 and 2'. This balance will give the following result:

Station 1 \(\rightarrow\) 1, 1', 3, 3', 2
Station 2 \(\rightarrow\) 2', 4, 5, 5', 6, 6'

In this case the idle times of the first and second stations will be 3 and 2, respectively, and the line efficiency will be 0.92. The solution meets the requirements for 1 and 1', 3 and 3', 5 and 5', 6 and 6'. This solution saves one station. It has also better efficiency while ignoring the grouping requirement for 2 and 2'.

The main idea behind the following formulation is balancing a mixed model assembly line by meeting the grouping requirements as much as possible, but also by avoiding the extra stations which may be needed to strictly meet these requirements.

4.2 MATHEMATICAL FORMULATION

The methodology is based on converting an MALE to a single product ALE. The first step of conversion is the arrangements of the task times of different products, and then transforming them to standard times by multiplying the time required to complete task i of product k by the number of product k to be produced within a shift time T.

Using the notation described in Chapter 3 the standard
Task time is defined as:

\[ t^{*}(i,k) = n(k) \cdot t(i,k) \quad k=1,2,\ldots,N \]
\[ i=1,2,\ldots,k(k) \]

In the second step the precedence diagrams of different products are combined together. They are then considered as a single item flow with the new task times \( t^{*}(i,k) \).

In the third step the grouping of similar tasks of different products is accomplished. The grouping requirement of these similar tasks can be either compulsory without the consideration of some extra stations, or relaxed from the assignment to the same station if it does not affect the number of stations. The tasks in the first category are combined together and will be treated as one single task. The tasks in the second category will be left out as they are and will be forced to be assigned to same stations during the solution procedure.

The fourth step is the omitting of the item subscript \( k \)'s and the renaming of all tasks of the final network. During this step the renaming procedure will be done in such a way that the identifying number of a task will always be greater than that of its predecessor(s); in other words, if "p" precedes "q", then \( p < q \).

The following variables will be utilized in this model:

\[ x(i,j) = \begin{cases} 
1 & \text{if task } i \text{ is assigned to station } j \\
0 & \text{otherwise}
\end{cases} \]
N = total number of product types
M. = total number of stations
J = total number of tasks
F(i) = subset of all tasks succeeding task i
P(i) = subset of all tasks preceding task i
E(i) = earliest station where task i can be assigned
L(i) = latest station where task i can be assigned
t(i) = time required for task i
T = shift time
C(i,j) = cost of assigning task i to station j
IP(i) = subset of all immediate predecessors of task i
G(y) = set of tasks of yth group
GSy(j) = subset of G(y), set of tasks which belong to group y but which have been scheduled to the jth station in the current partial solution
gsy(j) = cardinality of set GSy(j)
GT = total number of groupings

Task grouping:

If the assignment of some tasks to the same station is desired without increasing the number of stations or affecting the optimal solution, a task grouping procedure is suggested as follows: The tasks which are desired to be assigned to the same station form a group, and if one of these tasks is scheduled to a station, the cost of scheduling the remaining tasks (of the same group) to that specific station is much less than the cost.
of scheduling them to another station. For instance, if tasks 
(i, j, k) belong to group "l" (G(l)={i, j, k}), and at some step of 
the algorithm, task i is assigned to station "t", then:

\[ C(t, j) < C(m, j) \quad m = 1, 2, \ldots, t-1, t+1, \ldots, M \]

\[ C(t, k) < C(m, k) \quad m = 1, 2, \ldots, t-1, t+1, \ldots, N \]

The above grouping is feasible if the intersection of the 
assignable station sets (AS) of elements which belong to the 
same group is not empty, and if the total time required to 
perform the tasks of a group does not exceed the shift time.
The assignable station set for task i is:

\[ AS(i) = \{ E(i), E(i)+1, E(i)+2, \ldots, L(i) \} \quad E(i) < L(i) \]

In this particular case the following must hold:

\[ AS(i) \cap AS(j) \cap AS(k) \neq \emptyset \quad \text{and} \quad \sum_{j \in G(l)} t(j) \leq T \]

The total costs of grouped tasks will be discussed later.

**Variable elimination:**

The variable elimination procedure of this problem is
similar to that of the formulation presented in Chapter 3. The
only difference lies in the omission of the superscripts and
n(k)'s which were already incorporated during the transformation
to standard times. Thus,

\[ E(i) = \begin{cases} 
1 \text{ if } \sum_{j \in P(i)} t(j) = 0 \\
[(t(i) + \sum_{j \in P(i)} t(j))/T] + \text{ otherwise}
\end{cases} \]
\[
L(i) = \begin{cases} 
M & \text{if } t(i) + \sum_{j \in S(i)} t(j) = 0 \\
M + 1 - [(t(i) + \sum_{j \in S(i)} t(j))/T] & \text{otherwise}
\end{cases}
\]

**Occurrence constraints:**

The constraints which assure occurrences of all tasks and interfere the split of tasks among stations are expressed as:

\[
\sum_{j = 1, 2, \ldots, J} X(i, j) \leq 1
\]

(4.1.a)

and

\[
\sum_{i = 1}^{J} \sum_{j = E(i)}^{L(i)} X(i, j) \geq J
\]

(4.1.b)

Equation (4.1.b) indicates that the sum of all variables in the model will at least be equal to the number of tasks; in other words, all tasks will be assigned. Equation (4.1.a) ensures that the sum of all variables pertaining to a particular task can not exceed one; that is, this task can not be assigned to more than one station at the same time.

**Shift time constraints:**

These are the constraints which guarantee that a station load can not exceed the given shift time.

\[
\sum_{i \in A T j} t(i) X(i, j) \leq T \quad j = 1, 2, \ldots, N
\]

(4.2)
where:

ATj is the set of tasks that can be assigned to station j
and ATj(k) is the kth element of this set.

Precedence constraints:

The precedence relations in Chapter 3 and also in
existing literature are expressed as one constraint for every
precedence relation.

The present formulation requires only (J-JS)
constraints to define all precedence relations in an assembly
line, where J is the total number of tasks and JS is the total
number of starting tasks (which do not have any predecessor).
Explicitly, only one constraint is needed for every task having a
predecessor, whatever the number of its predecessors.

Considering that a job network has at least (J-JS)
precedence relations, the use of the present formulation allows
all precedence relations to be expressed with a minimum number
of constraints.

If a specific task \( t \) has \( p \) immediate predecessors
(IP(t)=1,2,...,p) the corresponding precedence constraint for
task \( t \) and all its predecessors will be:

\[
\alpha(1,1)X(1,1) + \alpha(1,2)X(1,2) + \ldots + \alpha(M,1)X(M,1) + \alpha(1,2)X(1,2) \\
\alpha(2,2)X(2,2) + \ldots + \alpha(M,2)X(M,2) + \ldots + \alpha(1,p)X(1,p) + \\
\alpha(2,p)X(2,p) + \ldots + \alpha(M,p)X(M,p) > \alpha(1,t)X(1,t) + \alpha(2,t)X(2,t) \\
+ \ldots + \alpha(M,t)X(M,t)
\] (4.4)
where:

\[
\alpha(i, j) = \begin{cases} 
M-1 \sum_{j=1,2,...,M-1} \sum_{i=1,2,...,p} p(1 + \sum_{k=j}^{M-1} p) & 
\text{if } j = 1, 2, \ldots, M-1 \\
1 & 
\text{if } j = M \\
L(i) C(i,j) & 
\text{if } j \in \mathcal{Z}(i)
\end{cases}
\]

Note that \(\alpha(i, j) = \alpha(2, j) = \ldots = \alpha(p-j)\) for \(j = 1, 2, \ldots, N\).

These above constraints ensure that a task \(t\) can be assigned to a station if and only if all its predecessors are assigned to the same or earlier stations.

The proof for equation (4.4) is given in Appendix D.

**Objective function:**

The objective function of the model is conceptually Bowman's[2] cost explosion function, but it is modified considerably in order to meet the requirements of the model.

The objective function is defined as:

\[
\min \sum_{i=1}^{J} \sum_{j \in \mathcal{Z}(i)} C(i,j)X(i,j) 
\]

where:
\[ C(i,j) = \frac{N \cdot g_{sy}(j) (j-M_0-1) - 2 + 2 \cdot t(i) \left( \sum_{k \in F} t(k) + 1 \right)}{2 + 2 \cdot t(i) \left( \sum_{k \in F} t(k) + 1 \right)} / 2 \] 

where:

\[ F = \{ i \mid \text{task } i \text{ does not precede any other element} \} \]

(set of final tasks)

\[ M_0 = \text{minimum number of stations (referring to 3.14)} \]

The idea behind this objective function is simply forcing the tasks of the same group to the same station, as well as minimizing the number of stations.

If a task does not belong to any group and it is not a final task, then its objective function coefficient is simply 2 for all stations; if it is a final one this coefficient is valid.
only for the stations earlier than Mo (see equation 4.5.a).

A task which is part of a group, say "y", has the same objective function value for stations 1,2,...,Mo regardless if whether it is a final task or not. This objective function coefficient is determined by gsy(i), the number of tasks of the same group that have been assigned to station i (see equation 4.5.b). The same equation is valid for the tasks which are not final and which are considered for the stations Mo+1,...,N.

If a final task is considered for stations later than Mo, a new factor is added in order to make it more costly to open a new station (see equations 4.5.c, 4.5.d). If this final task belongs to a group, the objective function coefficient will again depend on gsy(i) (see equation 4.5.d).

As an illustration, consider a three-item line balancing problem, where there is a grouping among the tasks (1,m,n) and (1,m) have been assigned to station j, and (1,m,n) ∈ F.

Hence Gs={1,m,n}

Gsy(j) = {1,m}: gsy(j) = 2 ; N = 3

If task n is attempted to be assigned to station "j",

\[ C(j,n)=\frac{2}{2}=2 \]

But for the same task the cost of assigning it to a different station is still \[ 2 \times 3 = 6 \]. Therefore, it becomes more attractive for task n to be assigned to station j.

The objective function terms are expressed in multiples of 2. The reason behind this is to avoid the remainder terms when they are divided throughout the algorithm. Since they are
always divided by multiples of 2, the results are always integer.

The assumption behind augmenting these objective function coefficients to the power \( N \) is that any of the groups \( (G(y), y=1,2,\ldots,G) \) can contain at most \( N \) elements. If more than \( N \) elements are assigned to any of the groups, \( N \) must be substituted by this number.

4.3 SOLUTION PROCEDURE

The NALB problem presented in this chapter has been solved using Geoffrion's[5] algorithm. Some steps in the enumeration process are strengthened in the sense of fathoming a partial solution, or shortened because of the structure of the mathematical model. These modifications are quite similar to those proposed by Thangavelu and Shetty[25], although they are slightly different due to the dynamic nature of the objective function, the grouping of tasks, and the fathoming criterion related to the latest assignable station of tasks.

Although it is assumed that Geoffrion's algorithm is known, some concepts and necessary explanations are given:

The algorithm basically serves two essential purposes:

1) Augmenting the partial solution whenever it may yield a feasible solution better than the incumbent one.

2) Backtracking whenever a feasible solution is better than the incumbent, or whenever a better solution is impossible
with the current partial solution.

An ordered set \( S \) which contains non zero elements will be used. If a particular variable \( X(j) \) is assigned its corresponding notation in \( S \) will be \(-j\) if \( X(j) = 0 \), and \(+j\) if \( X(j) = 1 \). Thus \( S = \{ 7, 9, -4 \} \) implies that \( X(7) = 1 \), \( X(9) = 1 \), \( X(4) = 0 \).

The following sequence of variables also will be used in the algorithm.

\[
S = X[1, AT(1)]; \quad X[1, AT(2)]; \quad \ldots ; X[i, AT(1)]; \quad X[i, AT(2)]; \quad \ldots ; X[M, AT(1)]; \quad X[M, AT(2)]; \quad \ldots
\]

Before exploring the solution procedure the following simplifications of the model will be described:

The constraints which guarantee the assignment of a task to only one station (see equation 4.1.b) are completely removed and are substituted by a simple rule in the augmentation phase. When one of the tasks is set to a station, all the other variables corresponding to this task will be augmented to zero. That is, if \( X(i_0, j_0) = 1 \), then \( X(i_0, j) = 0 \) for all \( j \neq j_0 \).

The coefficients of all variables of the constraint which assure the occurrence of tasks (equation 4.1.b) are the same and are equal to 1. Therefore, in the computer program, this constraint is expressed by a counter (unsubscripted variable) rather than an array. Whenever a variable is assigned to a station, this counter is augmented by 1 and whenever an assignment of a task is removed from current partial solution, it is decreased by 1.
The shift time constraints are substituted by counters. If a partial solution is augmented by \( X(i,j) \), the time counter of station \( i \) is augmented by the required time of task \( j \). If \( X(i,j) \) is removed from current partial solution this counter is decreased by \( t(j) \).

The augmentation phase of the algorithm is the assignment of the leftmost, unassigned variable of sequence \( \mathcal{S} \). If the assignment of this variable is feasible, set \( \mathcal{S} \) will be augmented by this variable with a positive sign. If it is not feasible it will also be augmented by the same variable but with a negative sign, and, furthermore this variable will be underlined. If the new entering variable belongs to the next station of the previously entered variable, an upper bound test (described in step 5 of the algorithm) must be performed. If the new entering variable with the positive sign belongs to a latest station of a variable, it will also be underlined.

The backtracking phase consists of finding the rightmost variable which is not underlined and changing its sign in order to find a better sequence than the incumbent one.

The following is the detailed algorithm. The flowchart of the program is given in Appendix A.

Step 0: Read the number of tasks, task identifiers, number of stations, shift time, task times, groups. Calculate the precedence matrix (using equation 4.4), and the objective function coefficients (using equation 4.5 and setting all
gsy(j) = 0, y = 1, 2, ..., GT and j = 1, ..., M. Set \( io = 1, \ jo = 1, \ Zs = 0, \ ZBAR = \phi \)

**Step 1:** Check if all tasks are assigned. If yes go to step 8, if not proceed to the next step.

**Step 2:** Select the leftmost unassigned variable of sequence \( \cup \), say \( X(i_1, j_1) \). If \( j_1 > j_o \) go to step 5. Otherwise set \( io = i_1, j_o = j_1 \) and go to next step.

**Step 3:** If \( io \in G(y), y = 1, 2, ..., GT \) go directly to the next step. If \( io \notin G(y), y = 1, 2, ..., GT \) find current \( gsy(jo) \) and \( C(io, jo) \), then proceed to next step.

**Step 4:** If \( Zs + C(io, jo) > ZBAR \) or \( tим(jo) + t(io) > T \) or corresponding precedence constraint defined in equation (4.3) is violated augment \( S \) by \( -X(io, jo) \), underline it and go to step 2. If not, augment \( S \) by \( +X(io, jo) \) (if \( j_o = L(io) \), underline it) and by \( -X(io, j) \) all the other variables for \( j \neq j_o \) and \( j \in ASC(io) \) then go to step 1.

**Step 5:** Compute \( j_2 = j_o + \left( \sum_{i} t(i) / T \right) + \) for \( X(i,j) = 1 \) and for all \( j \). If \( j_2 \) is greater than \( M \) go to step 7. Otherwise proceed to next step.

**Step 6:** Compute \( Zest = Zs + C(f, j_2) \)

where: \( t(f) = \min_{i \in F} t(i) \)
If Zest is greater than ZBAR go to step 7. If not set j=j1 and go to step 2.

Step 7: Select the rightest variable which is not underlined. If none exists stop. Otherwise multiply this variable by -1, drop all elements to the right, update the sets G(y), y=1,2,...,CT, gsy(j), j=1,2,...,N and go to step 2.

Step 8: If Zs > ZBAR go to step 7. If not, set Sc=S, ZBAR=Zs and go to step 7.

An illustration of the algorithm is given by means of a small problem the flow diagram of which is given in Figure 4.1.

MATHEMATICAL MODEL:
T=15, N=2, M=3, No=2, G(1)={2,3}
E(1)=1, L(1)=2; E(2)=1, L(2)=3; E(3)=1, L(3)=3; E(4)=2, L(4)=3
S = (X(1,1), X(2,1), X(3,1), X(1,2), X(2,2), X(3,2), X(4,2), X(2,3), X(3,3), X(4,3))

Occurrence constraints:
X(1,1) + X(2,1) + X(3,1) + X(1,2) + X(2,2) + X(3,2) + X(3,2) + X(4,2) + X(2,3) + X(3,3) + X(4,3) ≥ 4
X(1,1) + X(1,2) ≤ 1
X(2,1) + X(2,2) + X(2,3) ≤ 1
X(3,1) + X(3,2) + X(3,3) ≤ 1
X(4,2) + X(4,3) ≤ 1
Precedence diagram of the sample problem
Figure (4.1)
Shift time constraints:
\[ 6X(1,1) + 5X(2,1) + 5X(3,1) \leq 15 \]
\[ 5X(2,2) + 5X(3,2) + 6X(4,2) \leq 15 \]
\[ 5X(2,3) + 5X(3,3) + 6X(4,3) \leq 15 \]

Precedence constraints:
\[ 3X(1,1) + 2X(1,2) \geq 3X(2,1) + 2X(2,2) + X(2,3) \]
\[ 3X(1,1) + 2X(1,2) \geq 3X(3,1) + 2X(3,2) + X(3,3) \]
\[ 7X(2,1) + 5X(2,2) + X(2,3) + 7X(3,1) + 5X(3,2) + X(3,3) \geq 10X(4,2) + 2X(4,3) \]

Objective function:
\[ z = 2X(1,1) + 2X(1,2) + 4/2 \cdot g_{x(1)}(1) + 4/2 \cdot X(2,1) + 4/2 \cdot g_{x(2)}(2) + 4/2 \cdot g_{x(3)}(3) + 4/2 \cdot X(2,3) + 4/2 \cdot g_{x(1)}(1) + 4/2 \cdot X(3,1) + 4/2 \cdot g_{x(2)}(2) + 4/2 \cdot X(3,2) + 4/2 \cdot g_{x(3)}(3) + 2X(4,2) + 28X(4,3) \]

SOLUTION METHOD:

io=jo=1, ZBAR=∅

\[ X(1,1) \cdot C(1,1)=2, \text{ feasible} \]
\[ S = \{ X(1,1), -X(1,2) \} \]
\[ Zs=2, \text{ time}(1)=6 \]

\[ X(2,1) \cdot C(2,1)=4, \text{ feasible} \]
\[ S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3) \} \]
\[ Zs=6, \text{ time}(1)=11, g_{x(1)}(1)=1 \]
\[ X(3,1) \text{ not feasible} \]

\[
S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1) \}
\]

\[ X(3,2) ji>jo, j2=2, Zest<ZBAR, feasible, C(3,2)=4 \]

\[
S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1), X(3,2), -X(3,3) \}
\]

gsl(2)=1, Zs=10, time(2)=5, jo=2

\[ X(4,2) C(4,2), feasible \]

\[
S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1), X(3,2), -X(3,3), X(4,2), -X(4,3) \}
\]

Zs=12, time(2)=11

All tasks are assigned. ZBAR=Zs, Sc=S, fathem.

\[
S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1), X(3,2), -X(3,3), -X(4,2) \}
\]

Zs=10, time(2)=5, jo=2

\[ X(4,3) C(4,3)=28 \]

Zs=38, ZS>ZBAR, fathem.

\[
S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1), -X(3,2) \}
\]

time(2)=0, gsl(2)=0, Zs=6, jo=2

\[ X(4,2) \text{ not feasible} \]
\[ s = \{ x(1,1), -x(1,2), x(2,1), -x(2,2), -x(2,3), -x(3,1), -x(3,2), -x(4,2) \} \]

\[ \rightarrow x(3,3) \]

\[ j_1 > j_2 = 3, \: Zest = 34, \: Zest > ZBAR, \: \text{fathom}. \]

\[ s = \{ x(1,1), -x(1,2), -x(2,1) \} \]

\[ t_{ime}(1) = 6, \: t_{ime}(2), \: t_{ime}(3) = 0, \: Zs = 2, \: j_c = 1 \]

\[ \rightarrow x(3,1) \: C(3,1) = 4, \: \text{feasible} \]

\[ s = \{ x(1,1), -x(1,2), -x(2,1), x(3,1), -x(3,2), -x(3,3) \} \]

\[ gs(t)(1) = 1, \: t_{ime}(1) = 11, \: Zs = 6 \]

\[ \rightarrow x(2,2) \: j_1 > j_c, \: j_2 = 3, \: Zest = 34, \: Zest > ZBAR, \: \text{fathom}. \]

\[ s = \{ x(1,1), -x(1,2), -x(2,1), -x(3,1) \} \]

\[ Zs = 2, \: gs(t)(1) = gs(t)(2) = 0, \: t_{ime}(1) = 6, \: j_c = 1 \]

\[ \rightarrow x(2,2) \: j_1 > j_c, \: j_2 = 3, \: Zest = 30, \: Zest > ZBAR, \: \text{fathom}. \]

\[ s = \{ -x(1,1) \} \]

\[ \rightarrow x(2,1) \: \text{not feasible} \]

\[ s = \{ -x(1,1), -x(2,1) \} \]

\[ \rightarrow x(3,1) \: \text{not feasible} \]

\[ s = \{ -x(1,1), -x(2,1), -x(2,1) \} \]
\[ \rightarrow X(1,2) \quad j_1 > j_0, \text{Zest} \rightarrow \text{ZEAR}, \text{fathom}. \]

\[ S = \{ \emptyset \}, \text{stop}. \]

4.4 OPERATIONAL RESULTS

The assembly line procedure described here is programmed in Fortran language to be run on an IBM 370/3031 computer. The flowchart of the program is given in Appendix A. The completely documented program can be found in Appendix B1.

Five sample programs are used in testing the present procedure. They are:

1) The problem given in Figure 1.1 (Test problem 1)
2) The problem due to Bowman[2] (Test problem 2)
3) The problem due to Moodie and Mandeville[19]
   (Test problem 3)
4) The problem due to Ignall[10] (Test problem 4)
5) The problem due to Moodie and Young[20]
   (Test problem 5)

Precedence diagrams and element times of these problems are given in Appendix C.

Since the last four problems are actually single item ALB problems, they are either modified and considered for multiproduct flows by adding additional item flows on the same lines or it is assumed that they already contain mixed model flows.

In order to test this procedure, these sample problems
are run first without considering the grouping requirements. In
the second runs the tasks of the same groups are forced to be
assigned to the same stations. It has been observed that when
these similar tasks are forced to be assigned to the same
station, the procedure forced them as much as possible. It can
be seen that forcing the broken groups (the group of tasks which
are forced but not assigned to the same stations) in order to
assign their tasks to the same station will end up with an extra
station requirement. In these problems the number of products
to be produced within a shift time are assumed 1 (m(k)=1,
k=1,2,...,N).

Smoothness index (SI) of solutions are also given. This
index can be defined as the square root of the sum of squares of
the time deviations from the maximum work time of each of the
stations in the balance. A smoothness index of zero indicates a
perfect balance.

\[
\text{Smoothness index} = \sqrt{\sum_{k=1}^{N} (S_{\text{max}} - S(k))^2}
\]

where: S(k) is the station time of station k.

\[
S_{\text{max}} = \text{Max}\{ S(1), S(2), ..., S(N) \}
\]

The computer times (CT) given for solutions with
grouping requirements, exclude input and output times; and are in
seconds.
TEST PROBLEM 1

Required shift time = 25

Without grouping:

| Station 1 | 1 2 3 5 7 | Slack time= 2 |
| Station 2 | 4 6 8 | Slack time= 2 |
| Station 3 | 9 10 11 | Slack time= 16 |

SI= 14.00

With grouping:

Grouping requirements:

| Group 1 | 1 2 |
| Group 2 | 3 6 |
| Group 3 | 4 5 |
| Group 4 | 8 9 |
| Group 5 | 10 11 |

Station assignments:

| Station 1 | 1 2 4 5 | Slack time= 2 |
| Station 2 | 3 6 7 | Slack time= 8 |
| Station 3 | 8 9 10 11 | Slack time= 10 |

SI= 10.00  CT= 0.057

TABLE 4.1
TEST PROBLEM 2

Required shift time = 20

Without grouping:

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>Slack time = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 2</td>
<td>3</td>
<td>0</td>
<td>Slack time = 3</td>
</tr>
<tr>
<td>Station 3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Station 4</td>
<td>6</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>Station 5</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Station 6</td>
<td>9</td>
<td>11</td>
<td>Slack time = 5</td>
</tr>
</tbody>
</table>

SI = 11.66

With grouping:

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Group 3</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Group 4</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Group 5</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>Slack time = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Station 3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Station 4</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Station 5</td>
<td>10</td>
<td>11</td>
<td>Slack time = 9</td>
</tr>
<tr>
<td>Station 6</td>
<td>12</td>
<td>13</td>
<td>Slack time = 7</td>
</tr>
</tbody>
</table>

SI = 10.00  CT = 0.480

TABLE 4.2
TEST PROBLEM 3

Required shift time = 15

Without grouping:

Station assignments:

| Station 1 | 1 2 3 | Slack time= 3 |
| Station 2 | 4 5 6 7 | Slack time= 0 |
| Station 3 | 8 9 10 | Slack time= 1 |
| Station 4 | 11 12 13 | Slack time= 1 |

SI = 3.32

With grouping:

Grouping requirements:

| GROUP 1 | 1 2 |
| GROUP 2 | 3 7 |
| GROUP 3 | 9 10 |
| GROUP 4 | 12 13 |

Station assignments:

| Station 1 | 1 2 4 | Slack time= 2 |
| Station 2 | 3 5 6 7 | Slack time= 1 |
| Station 3 | 8 9 10 | Slack time= 1 |
| Station 4 | 11 12 13 | Slack time= 1 |

SI = 1.00  CT = 1.501

TABLE 4.3
TEST PROBLEM 4

Required shift time = 10

Without grouping:

Station assignments:

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>Slack time= 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 2</td>
<td>6</td>
<td>8</td>
<td>Slack time= 2</td>
<td></td>
</tr>
<tr>
<td>Station 3</td>
<td>3</td>
<td>10</td>
<td>Slack time= 0</td>
<td></td>
</tr>
<tr>
<td>Station 4</td>
<td>4</td>
<td>7</td>
<td>Slack time= 0</td>
<td></td>
</tr>
<tr>
<td>Station 5</td>
<td>9</td>
<td>11</td>
<td>Slack time= 1</td>
<td></td>
</tr>
</tbody>
</table>

SI= 2.45

With grouping:

Grouping requirements:

<table>
<thead>
<tr>
<th>GROUP 1</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP 2</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

Station assignments:

| Station 1 | 1 | 2 | Slack time= 2 |
| Station 2 | 5 | 6 | 8 | Slack time= 1 |
| Station 3 | 3 | 10 | Slack time= 0 |
| Station 4 | 4 | 7 | Slack time= 0 |
| Station 5 | 9 | 11 | Slack time= 1 |

SI= 2.45  CT= 0.980

TABLE 4.4
TEST PROBLEM 5

Required shift time = 22

Without grouping:

Station assignments:

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Slack time=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
<td>Slack time=1</td>
</tr>
<tr>
<td>Station 3</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Station 4</td>
<td>14</td>
<td>15</td>
<td>18</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>Station 5</td>
<td>16</td>
<td>17</td>
<td>20</td>
<td></td>
<td>Slack time=3</td>
</tr>
</tbody>
</table>

SI = 3.32

With grouping:

Grouping requirements:

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 2</td>
<td>8</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Group 3</td>
<td>13</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Group 4</td>
<td>16</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Group 5</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Station assignments:

<table>
<thead>
<tr>
<th>Station 1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Slack time=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
<td>Slack time=1</td>
</tr>
<tr>
<td>Station 3</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>Station 4</td>
<td>11</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>Station 5</td>
<td>16</td>
<td>17</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

SI = 0.0  CT = 1.652

TABLE 4.5
CHAPTER 5

COST ASPECTS OF THE PROBLEM

5.1 DEFINITION OF COSTS

In the first part of this study the main objective was the optimization of a mixed-model assembly line by minimizing the number of stations and maximizing line efficiency as well. When similar tasks of different products have been considered for different stations, the only objective was a high line efficiency.

Another factor which can be incorporated into this research is the cost aspect of the problem. Handling the grouping requirements from the cost point of view is achieved through the incorporation of two types of cost. They are the cost of duplicating the equipment, and the cost of opening a new station. In Chapter 4, if the assignment of similar tasks to the same station would cause an extra station, this assignment would not have been made. Although in most cases a new station is not preferred, consideration of the cost of splitting similar tasks is of importance to the problem as well. For example, if the reason behind a specific grouping is the use of the same equipment, one should also consider the cost of duplicating this equipment for more than one station. If the assignment of all tasks of this group to the same station will require a new station, then the cost of the new station should also be
considered and compared with the duplication cost.

The assumptions on these costs can be explained as follows:

The equipment duplication cost is incurred if and only if a group is split. It also depends on the number of stations this group is split into. There is no duplication cost if a specific task of a group is assigned to a station, while the other tasks of the same group have not been assigned to any station. If a task of this group is assigned to a station to which another task of the same group had been assigned, there will be no cost occurrence either. The duplication cost prevails when the second task of this group is assigned to a different station than the one to which the first task was already assigned. For example, if there is a group "1" made up of 3 tasks \( G(1) = \{i,j,k\} \) and we have 3 stations, the following tableau can give the cost figures for different cases. The columns represent the 3 stations \( (s = 1,2,3) \) and the rows represent different alternatives of assigning the tasks of group "1" to these stations.

<table>
<thead>
<tr>
<th></th>
<th>s=1</th>
<th>s=2</th>
<th>s=3</th>
<th>Total dup. cost due to group &quot;1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i,j,k )</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>( i )</td>
<td>( j )</td>
<td></td>
<td></td>
<td>( C_d(1) )</td>
</tr>
<tr>
<td>( i )</td>
<td>( j )</td>
<td>( k )</td>
<td></td>
<td>( 2C_d(1) )</td>
</tr>
<tr>
<td>( i )</td>
<td>( j,k )</td>
<td></td>
<td></td>
<td>( C_d(1) )</td>
</tr>
<tr>
<td>( i,j,k )</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
In this tableau \( C_d(1) \) is the duplication cost of tasks belonging to group "1". The cost of opening a new station is incorporated into the problem as the second cost. This cost prevails at every station additional to the minimum number of stations beyond \( N_0 \) (see equation 3.14). It in fact depends on how many more stations are added after \( N_0 \).

5.2 MATHEMATICAL FORMULATION

The incorporation of these costs is achieved through the modification of the objective function. In the new objective function, the variables corresponding to grouped tasks contain the cost of equipment duplication and the occurrence of these costs change upon the current partial solution.

The cost of a new station is incorporated only to the variables pertaining to the final task. The model assumes only one final task for the job network. In the cases where the line has more than one final task, one dummy task is needed. It must succeed all the other final tasks and have operation time zero.

Besides the notation and the sets defined in Chapter 4, the following expressions are also necessary in order to express the new objective function:

\[
Cst = \text{cost of opening a new station}
\]

\( C_d(y) = \text{cost of duplication due to group } y \)

\( g_t(y) = \text{Number of tasks which belong to group } y \) but which have been assigned to different stations
in the current partial solution.

\[ f = \text{final task} \]

Finally the objective function is:

\[
\sum_{i=1}^{N} \sum_{j \in E(i)} C(i,j) X(i,j) \quad (5.1)
\]

where:

\[
[1 - \min\{gsy(j), 1\}] [\min\{gt(y), 1\}] \cdot Cd(y) \quad (5.1.a)
\]

for
\[
j = 1, 2, \ldots, N
\]
\[
i \notin G(y), y = 1, 2, \ldots, GT
\]
\[
i \neq f
\]

\[
0 \quad (5.1.b)
\]

for
\[
j = 1, 2, \ldots, N
\]
\[
i \notin G(y), y = 1, 2, \ldots, GT
\]
\[
i \neq f
\]

\[
C(i,j) = \max\{j-Mo, 0\} Cst \quad (5.1.c)
\]

for
\[
j = 1, 2, \ldots, N
\]
\[
i \notin G(y), y = 1, 2, \ldots, GT
\]
\[
i = f
\]

\[
[1 - \min\{gsy(j), 1\}] [\min\{gt(y), 1\}] \cdot Cd(y) + \max\{j-Mo, 0\} Cst \quad (5.1.d)
\]

for
\[
j = 1, 2, \ldots, N
\]
\[
i \notin G(y), y = 1, 2, \ldots, GT
\]
\[
i = f
\]

Equation (5.1.b) indicates that in the objective
function if a task does not belong to any group, and if it is not a final task, the objective function coefficient for the corresponding variable will be zero regardless of the current partial solution.

If task \( j \) is not final but nevertheless belongs to a group the objective function value can be determined by the equation (5.1.a). If the task \( j \) belongs to group \( k \) and none of the elements of this group has been assigned \( (g_t(y)=0) \), \( C(i,j) \) would be zero due to the second bracketed term. But if one of the elements has been assigned, the second bracketed term would be 1 and the coefficient would be determined by the first bracketed term. If the previously assigned task has been assigned to a station other than \( i \), the first term would be 1 and the final result would be \( C(i,j)=C_d(y) \). If it has been assigned to station \( i \), the first term would be zero so that \( C(i,j) \).

Equations (5.1.c) and (5.1.d) deal with the final task. Equation (5.1.c) states that if the final task is not a part of any group and if it is considered for assignment to a station before \( x_0 \), the corresponding \( C(i,j) \) will be zero. If it is a candidate for a later station than \( x_0 \), its objective function coefficient will be determined by the station number after \( x_0 \). If a final task is part of a group, the coefficients corresponding to this task will be made up of the terms due to grouping and also due to the new station cost (equation 5.1.d). The present objective function is subject to the same
restrictions of the model of Chapter 4 (equations 4.1.a, 4.1.b, 4.2, 4.4).

5.3 SOLUTION PROCEDURE

The solution procedure presented in Chapter 4 can also be employed to solve this model with some minor modifications, which are mostly due to the new objective function. Since the objective function of the model has completely been changed, its coefficients must be calculated according to the equation (5.1). The algorithm of section 4.3 is completely valid for this new problem. The only minor modification is in Step 3 and Step 7 of the algorithm, where the new variables $gt(y), y=1, 2, \ldots, CT$ must be updated for every iteration throughout the algorithm.

The modified algorithm is illustrated by means of the small problem presented in Section 4.3. The mathematical model remains the same except for the objective function. The new objective function is:

$$Z = 100[1 - \min\{gsl(1), 1\}][\min\{gt(1), 1\}] X(2, 1) +$$
$$100[1 - \min\{gsl(1), 1\}][\min\{gt(1), 1\}] X(3, 1) +$$
$$100[1 - \min\{gsl(2), 1\}][\min\{gt(1), 1\}] X(2, 2) +$$
$$100[1 - \min\{gsl(2), 1\}][\min\{gt(1), 1\}] X(2, 2) +$$
$$100[1 - \min\{gsl(3), 1\}][\min\{gt(1), 1\}] X(2, 3) +$$
$$100[1 - \min\{gsl(3), 1\}][\min\{gt(1), 1\}] X(3, 3) + 50X(4, 3)$$

The new cost figures are: $Cc(1)=100$, $Cst=50$
SOLUTION METHOD

io=jo=1, ZBAR=0, gsl(1)=gsl(2)=gsl(3)=gt(1)=0

--> X(1,1) C(1,1)=0, feasible

S = { X(1,1), -X(1,2) }

Zs=0, time(1)=6

--> X(2,1) C(2,1)=0, feasible

S = { X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3) }

Zs=0, time(1)=1, gsl(1)=gt(1)=1

--> X(3,1) not feasible

S = { X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3),

-X(3,1) }

--> X(3,2) j1>jo, j2=2, Zest<ZBAR, feasible, C(3,2)=100

S = { X(1,1), -X(1,2), X(1,1), -X(2,2), -X(2,3),

-X(3,1), X(3,2), -X(3,3) }

gsl(2)=1, gt(1)=2, Zs=100, time(2)=5

--> X(4,2) C(4,2)=0, feasible

S = { X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3),

-X(3,1), X(3,2), -X(3,3), X(4,2), -X(4,3) }

Zs=100, time(2)=11

All tasks are assigned. ZBAR=Zs, Sc=S, fathom.

S = { X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3),
\(-X(3,1), X(3,2), -X(3,3), -X(4,2) \)  
\(t\text{ime}(2)=5\)

\(--\rightarrow X(4,3), C(4,3)=50, j1>j0, j2=3, Z\text{est}=Zs+50 > Z\text{BAR}, fathom.\)
\(S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1), -X(3,2), -X(4,2) \}\)
\(t\text{ime}(2)=0, g\text{s}1(2)=0, g\text{t}(1)=1, Zs=0\)

\(--\rightarrow X(4,2) \text{ not feasible}\)
\(S = \{ X(1,1), -X(1,2), X(2,1), -X(2,2), -X(2,3), -X(3,1), -X(3,2), -X(4,2) \}\)

\(--\rightarrow X(3,3) C(3,3)=100, j1>j0, j2=3, Z\text{est}=50 < Z\text{BAR}\)
\(Zs=Z\text{BAR}, fathom.\)
\(S = \{ X(1,1), -X(1,2), -X(2,1) \}\)
\(g\text{s}1(1)=g\text{s}1(2)=-, g\text{t}(1)=0, t\text{ime}(1)=6, t\text{ime}(2)=0, Zs=0\)

\(--\rightarrow X(3,1) C(3,1)=0, \text{ feasible}\)
\(S = \{ X(1,1), -X(1,2) -X(2,1), X(3,1), -X(3,2), -X(3,3) \}\)
\(t\text{ime}(1)=11, g\text{s}1(1)=1, g\text{t}(1)=1, Zs=0\)

\(--\rightarrow X(2,2) j0>j1 \ j2=2 \ C(2,2)=100\)
\(Zs=100=Z\text{BAR}, fathom.\)
\(S = \{ X(1,1), -X(1,2), -X(2,1), -X(2,2), -X(2,3) \}\)
\(t\text{ime}(1)=6, g\text{s}1(1)=0, g\text{t}(1)=0\).
\[\rightarrow X(2,2) \text{ j1>j0, j2=3, Zest=50 < ZBAR, C(2,2)=0}\]
\[S = \{ x(1,1), -X(1,2), -X(2,1), -X(3,1), X(2,2),\]
\[-X(2,3) \}\]
\[gt(1)=1, gsl(2)=1, Zs=0, time(1)=6, time(2)=5\]

\[\rightarrow X(3,2) \quad C(3,2)=0 \text{ feasible}\]
\[S = \{ x(1,1), -X(1,2), -X(2,1), -X(3,1), X(2,2),\]
\[-X(2,3), X(3,2), -X(3,3) \}\]
\[time(2)=10, gsl(3)=2, gt(1)=2.\]

\[\rightarrow X(4,2) \text{ not feasible}\]
\[S = \{ x(1,1), -X(1,2), -X(2,1), -X(3,1), X(2,2),\]
\[-X(2,3), X(3,2), -X(3,3) -X(4,2) \}\]

\[\rightarrow X(4,3) \text{ j1>jc, j2=3, Zest=50 < ZBAR, C(4,3)=50}\]
\[S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), X(2,2),\]
\[-X(2,3), X(3,2), -X(3,3) -X(4,2), X(4,3) \}\]
\[time(3)=6.\]

All tasks are assigned, fathem.
\[S = \{ X(1,1), -X(1,2), -X(2,1) -X(3,1), X(2,2),\]
\[-X(2,3), -X(3,2) \}\]
\[Zs=0, time(1)=6, time(2)=5, time(3)=0, gsl(2)=1, gt(1)=0.\]

\[\rightarrow X(4,2) \text{ not feasible}\]
\[ S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), X(2,2), -X(2,3), -X(3,2), -X(4,2) \} \]

\[ \rightarrow X(3,3), j_1 > j_c, j_2 = 3, Zest = 50 = ZBAR, fathom. \]
\[ S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), -X(2,2) \} \]
\[ \text{time}(1)=6, \text{time}(2)=, gsl(2)=0, gt(1)=, Zs=0 \]

\[ \rightarrow X(3,2) \text{ feasible } C(3,2)=0 \]
\[ S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), -X(2,2), X(3,2), -X(3,3) \} \]
\[ \text{time}(2)=5, \text{gsl}(2)=1, gt(1)=1. \]

\[ \rightarrow X(4,2) \text{ not feasible } . \]
\[ S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), -X(2,2), X(3,2), -X(3,3), -X(4,2) \} \]

\[ \rightarrow X(2,3), j_1 > j_c, j_2 = 3, Zest = 50 = ZBAR, fathom. \]
\[ S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), -X(2,2), -X(3,2) \} \]
\[ \text{time}(2)=\text{time}(3)=0, \text{gsl}(2)=0, gt(1)=0. \]

\[ \rightarrow X(4,2) \text{ not feasible } \]
\[ S = \{ X(1,1), -X(1,2), -X(2,1), -X(3,1), -X(2,2), -X(3,2), -X(4,2) \} \]

\[ \rightarrow X(2,3) \text{ feasible, } j_1 > j_c, j_2 = 3, Zest = 50 = ZBAR, fathom. \]
\[ S = \{ -X(1,1) \} \]
time(1)=0.

---> X(2,1) not feasible.
S = { -X(1,1), -X(2,1) }

---> X(3,1) not feasible
S = { -X(1,1), -X(2,1), -X(3,1) }

---> X(1,2) feasible, j1>j, j2=3, Zest=50=ZBAR, fathcm.
S = { ∅ }, stop.

5.4 OPERATIONAL RESULTS

The modifications described above are incorporated to the Fortran program presented in Appendix B1. Although both of the solution procedures described in sections 4.3 and 5.3 are algorithmically the same, because of the changes in the calculation of the objective function coefficients, and due to the introduction of a new variable (gt(y), y=1,2,...,GT) these incorporations needed considerable alterations of program. Due to these alterations, the modified program has been given separately in Appendix B2.

The same test problems are used in testing this new procedure. These sample problems are run first without grouping, then with grouping and costs as well.

The costs are given considering the two different cases.
In the first case, the equipment duplication costs of similar tasks \( (C_{d(y)}, y=1, 2, \ldots, GT) \) are much more expensive than the new station cost \( (C_{st}) \). In second case, the cost of opening a new station is much higher than the duplication costs.

It can be observed that in both cases the algorithm chooses the most economical assignments for the given costs. If the duplication costs are greater than the costs of the new station, it strictly meets these grouping requirements and does not hesitate to open a new station. If they are less than the new station costs, the procedure does not open a new station but it still meets these grouping requirements as much as possible without creating a new station.
TEST PROBLEM 1

Required shift time = 30

Without grouping:

Station assignments:
| Station 1 | 1 2 3 4 5 | Slack time= 3 |
| Station 2 | 6 7 8 9 10 11 | Slack time= 2 |

SI= 1.00

With grouping:

Grouping requirements:
| Group 1 | 1 2 | Dup. Cost= 500 |
| Group 2 | 3 6 | Dup. Cost= 500 |
| Group 3 | 4 5 | Dup. Cost= 500 |
| Group 4 | 8 9 | Dup. Cost= 500 |
| Group 5 | 10 11 | Dup. Cost= 500 |

Case 1 ( Cst < Cd(y), y=1, ...5 )
Cost of new station = 100
Station assignments:
| Station 1 | 1 2 4 5 | Slack time= 7 |
| Station 2 | 3 6 7 8 9 | Slack time= 4 |
| Station 3 | 10 11 12 | Slack time= 24 |

SI= 20.22  CT= 1.213

Case 2 ( Cst > Cd(y), y=1, ...5 )
Cost of new station = 990
Station assignments:
| Station 1 | 1 2 3 4 5 | Slack time= 3 |
| Station 2 | 6 7 8 9 10 11 12 | Slack time= 2 |

SI= 1.00  CT= 0.103

TABLE 5.1
TEST PROBLEM 2

Required shift time = 25

Without grouping:

Station assignments:

| Station 1 | 1 2 4 6 | Slack time= 0 |
| Station 2 | 3 8    | Slack time= 0 |
| Station 3 | 5 7 10 12 | Slack time= 0 |
| Station 4 | 9 11 13 | Slack time= 0 |

SI= 0.0

With grouping:

Grouping requirements:

| Group 1 | 1 2 | Dup. Cost= 300 |
| Group 2 | 4 5 | Dup. Cost= 300 |
| Group 3 | 6 7 | Dup. Cost= 200 |
| Group 4 | 8 9 | Dup. Cost= 100 |
| Group 5 | 12 13 | Dup. Cost= 300 |

Case 1 (Cst < Cd(y), y=1,...,5)
Cost of new station = 50
Station assignments:

| Station 1 | 1 2 | Slack time= 5 |
| Station 2 | 3    | Slack time= 8 |
| Station 3 | 4 5 6 7 | Slack time= 6 |
| Station 4 | 8 9 11 | Slack time= 2 |
| Station 5 | 10 12 13 14 | Slack time= 4 |

SI= 8.06    CT= 5.516

Case 2 (Cst > Cd(y), y=1,...,5)
Cost of new station = 900
Station assignments:

| Station 1 | 1 2 4 6 | Slack time= 0 |
| Station 2 | 3 8    | Slack time= 0 |
| Station 3 | 5 7 10 12 | Slack time= 0 |
| Station 4 | 9 11 13 | Slack time= 0 |

SI= 0.0    CT= 0.479

TABLE 5:2
TEST PROBLEM 3

Required shift time = 12

Without grouping:

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Slack time = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>Slack time = 2</td>
</tr>
<tr>
<td>Station</td>
<td>7</td>
<td>10</td>
<td></td>
<td>Slack time = 2</td>
</tr>
<tr>
<td>Station</td>
<td>8</td>
<td>11</td>
<td>13</td>
<td>Slack time = 1</td>
</tr>
<tr>
<td>Station</td>
<td>12</td>
<td>11</td>
<td>13</td>
<td>Slack time = 1</td>
</tr>
</tbody>
</table>

SI = 3.00

With grouping:

<table>
<thead>
<tr>
<th>Group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>Slack time = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>Slack time = 2</td>
</tr>
<tr>
<td>Group</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>Slack time = 1</td>
</tr>
<tr>
<td>Group</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>Slack time = 0</td>
</tr>
</tbody>
</table>

Case 1 (Cost < C_d(y), y = 1, ..., 5)
Cost of new station = 50

Station assignments:

SI = 8.15  CT = 5.326

Case 2 (Cost > C_d(y), y = 1, ..., 5)
Cost of new station = 950

Station assignments:

SI = 3.32  CT = 1.605

TABLE 5.3
TEST PROBLEM 4

Required shift time = 10

Without grouping:

Station assignments:

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Slack time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Station 2</td>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Station 3</td>
<td>3</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Station 4</td>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Station 5</td>
<td>9</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

SI = 2.45

With grouping:

Grouping requirements:

<table>
<thead>
<tr>
<th>Group 1</th>
<th>5</th>
<th>6</th>
<th>Dup. Cost = 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 2</td>
<td>7</td>
<td>10</td>
<td>Dup. Cost = 500</td>
</tr>
</tbody>
</table>

Case 1 (Cst < Cd(y), y=1,2)

Cost of new station = 100

Station assignments:

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Slack time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Station 2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Station 3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Station 4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Station 5</td>
<td>7</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Station 6</td>
<td>9</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

SI = 4.00  CT = 2.456

Case 2 (Cst > Cd(y), y=1,2)

Cost of new station = $900

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Slack time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Station 2</td>
<td>3</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Station 3</td>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Station 4</td>
<td>5</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

SI = 2.45  CT = 0.989

TABLE 5.4
TEST PROBLEM 5

Required shift time = 22

Without grouping:

Station assignments:

| Station 1 | 1  | 2  | 3  | 4  | Slack time = 1 |
| Station 2 | 5  | 6  | 7  |    | Slack time = 1 |
| Station 3 | 8  | 9  | 10 | 11 | 12 | 13 | Slack time = 0 |
| Station 4 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | Slack time = 0 |
| Station 5 | 16 | 17 | 20 |    |    |    |    |    | Slack time = 3 |

SI = 3.32

With grouping:

Case 1 (Cst < Cd(y), y=1, ..., 5)
Cost of new station = 100

Grouping requirements:

| Group 1 | 2  | 5  | Dup. Cost = 750 |
| Group 2 | 8  | 21 | Dup. Cost = 850 |
| Group 3 | 13 | 14 | Dup. Cost = 500 |
| Group 4 | 16 | 18 | Dup. Cost = 650 |
| Group 5 | 19 | 20 | Dup. Cost = 800 |

Station assignments:

| Station 1 | 1  | 3  | 4  |    | Slack time = 4 |
| Station 2 | 2  | 5  | 6  |    | Slack time = 6 |
| Station 3 | 7  | 8  | 21 |    | Slack time = 0 |
| Station 4 | 9  | 10 | 11 | 12 | 13 | 14 | Slack time = 4 |
| Station 5 | 15 | 16 | 18 |    |    |    |    | Slack time = 0 |
| Station 6 | 17 | 19 | 20 | 22 |    |    |    |    | Slack time = 4 |

SI = 12.85  CT = 45.583

Case 2 (Cst > Cd(y), y=1, ..., 5)
Cost of new station = 900

Grouping requirements:

| Group 1 | 2  | 5  | Dup. Cost = 50 |
| Group 2 | 8  | 21 | Dup. Cost = 50 |
| Group 3 | 13 | 14 | Dup. Cost = 50 |
| Group 4 | 16 | 18 | Dup. Cost = 50 |
| Group 5 | 19 | 20 | Dup. Cost = 50 |
**Station assignments:**

<table>
<thead>
<tr>
<th>Station</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Slack time= 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
<td>Slack time= 1</td>
</tr>
<tr>
<td>Station</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>Station</td>
<td>11</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>Station</td>
<td>16</td>
<td>17</td>
<td>19</td>
<td>20</td>
<td>22</td>
</tr>
</tbody>
</table>

SI = 0.0   CT = 1.719

**TABLE 5.5**
CHAPTER 6

CONCLUSIONS

A multiproduct assembly line balancing and the grouping requirements of similar tasks of different products are investigated in this study.

A general 0-1 integer linear programming model for mixed model assembly line balancing problem is developed in Chapter 3. The number of variables and constraints are much less than the similar formulations proposed in the literature.

In Chapter 4, the grouping requirements are relaxed. They are reconstructed as much as possible, preserving the same line efficiency as it was without grouping.

In Chapter 5, the cost aspects of the problem is introduced. The costs due to the repetition of similar tasks of different products in different stations and the cost of opening a new station are considered. These costs are incorporated into the solution procedure developed in Chapter 4 to establish the most economical station assignments.

The models suggested in Chapters 4 and 5 are tested by means of some test problems in the literature. It is observed that the results have been satisfactory in terms of grouping requirements and the incorporated costs. If the cost effect is not considered, grouping requirements are met as much as possible without increasing the number of stations; if the costs
are included, the problem is solved on the basis of the minimum cost criterion.
APPENDIX A
PROOF OF EQUATION (4.4)
Let a specific task \( t \) have \( p \) predecessors and the total number of stations under consideration is \( M \). Then the corresponding precedence equation will be:

\[
\alpha(1,1)X(1,1) + \alpha(1,2)X(1,2) + \ldots + \alpha(M,1)X(M,1) + \alpha(1,2)X(1,2) \\
\alpha(2,2)X(2,2) + \ldots + \alpha(M,2)X(M,2) + \ldots + \alpha(1,p)X(1,p) + \\
\alpha(2,p)X(2,p) + \ldots + \alpha(M,p)X(M,p) \geq \alpha(1,t)X(1,t) + \alpha(2,t)X(2,t) \\
+ \ldots + \alpha(M,t)X(M,t) \\
\]  
\( (A.1) \)

It is required that if one of the predecessors is assigned to station \( m \), then the earliest station for task \( t \) is \( m \): Once a task \( i \) (\( i \in IP_t \)) is assigned to station \( m \), whatever the maximum value of the left hand side of the equation is, this constraint must NOT be satisfied if task \( t \) is assigned to an earlier station than \( m \). This can be expressed as:

\[
\alpha(\pi,i) + \sum_{j=1\atop j \neq i}^{p} \max\{\alpha(1,j), \alpha(2,j), \ldots, \alpha(\pi,j)\} < \alpha(\pi-1,t) \\
\]  
\( (A.2) \)

Considering \( \alpha(i,j) > \alpha(2,j) > \ldots > \alpha(M,j) \), \( (A.2) \) can be expressed as:

\[
\alpha(\pi,j) + \sum_{j=1\atop j \neq i}^{p} \alpha(1,j) < \alpha(\pi-1,t) \\
\]  
\( (A.3) \)
Given that \( \alpha(i, 1) = \alpha(i, 2) = \ldots = \alpha(i, p) \) for \( i = 1, 2, \ldots, N \) and \( \alpha(m-1, j) = p \alpha(m-1, j) \) for \( j \in \{1, p\} \) 

(A.2) can be rewritten as:

\[ \alpha(m, i) + (p-1) \alpha(1, j) < p \alpha(m-1, j) \quad \text{for} \quad i \geq 2 \quad (A.4) \]

Substituting \( \alpha(i, j) = 1 + \sum_{k=1}^{N-1} p \) \( \quad \text{for} \quad i < N-1 \)

and \( \alpha(N, j) = 1 \quad \text{for} \quad j = 1, 2, \ldots, p \)

into (A.4) yields:

\[ 1 + \sum_{k=m}^{m-1} p + (p-1) \left[ 1 + \sum_{k=1}^{N-1} p \right] < p \left[ 1 + \sum_{k=m-1}^{N-1} p \right] \quad (A.5) \]

With some algebraic manipulations:

\[ \sum_{k=m}^{N-1} p - \sum_{k=1}^{N-1} p + p \sum_{k=1}^{N-1} p < \sum_{k=m-1}^{N-1} p \]

\[ \sum_{k=1}^{N-1} p - \sum_{k=m}^{N-1} p + p \sum_{k=m-1}^{N-1} p - \sum_{k=1}^{N-1} p > 0 \]

\[ \sum_{k=1}^{m} p - \sum_{k=m}^{m-1} p > 0 \]

\[ \sum_{k=1}^{m-1} p - p \sum_{k=2}^{m-1} p > 0 \]

\[ p > 0 \quad \text{which is always true since} \quad p \quad \text{is always positive} \]

(number of predecessors)
APPENDIX B
FLOW CHART
APPENDIX C1

COMPUTER PROGRAM OF THE ALGORITHM DESCRIBED IN CHAPTER
C *-----------------------------------------------------------------------*
C * VARIABLES USED IN THE PROGRAM:                                     *
C * (THE VARIABLES GIVEN IN PARANTHSES INDICATE THE MINIMUM DIMENSIONS  *
C * REQUIRED FOR THE ARRAYS)                                            *
C *-----------------------------------------------------------------------*
C * A(I,J): ARRAY WHICH CONTAINS ALL PREDECESSORS/SUCCESSORS            *
C * OF TASK I (NTASK,NTASK)                                             *
C * ARRI(J): WORKING ARRAY (3,NACVAR)                                  *
C * B(I): RHS VECTOR (CREATED BY THE PROGRAM) (NTASK)                   *
C * BMIAT(I,J): WORKING VECTOR (NTASK)                                  *
C * C(I): OBJECTIVE FUNCTION COEFFICIENTS (CREATED BY THE              *
C * PROGRAM)                                                          *
C * CVAR: TOTAL NUMBER OF INDEPENDENT VARIABLES GENERATED BY           *
C * THE PROGRAM. CVAR DEPENDS ON THE NVAR, NTASK, TASK TIMES, TYPE OF *
C * NETWORK. A MAXIMUM VALUE THAT CVAR MAY TAKE IS (NTASK*NSTR)         *
C * EST(I): EARLIEST STATION WHERE TASK I CAN BE ASSIGNED(NTASK)        *
C * FIV: CURRENT OBJECTIVE FUNCTION VALUE                              *
C * G(I): GROUP TO WHICH TASK I BELONGS TO (NTASK)                      *
C * GWAT(I,J): WORKING ARRAY (NTASK,NTASK)                             *
C * GROUP(I): WORKING ARRAY FOR PRINTING RESULTS (10)                  *
C * ICC: SHIFT TIME                                                    *
C * INDA(I): WORKING ARRAY (NTASK)                                     *
C * LAST(I): WORKING ARRAY (NTASK)                                     *
C * LOC(I): WORKING ARRAY (NTASK)                                      *
C * LST(I): LATEST STATION WHERE TASK I CAN BE Assigned (NTASK)        *
C * MAX(I,J): PRECEDENCE MATRIX (CREATED BY THE PROGRAM)               *
C * (NTASK-NSTR,CVAR)                                                  *
C * MAXSUC: NUMBER OF MAXIMUM IMMEDIATE SUCCESSOR                      *
C * NST: NUMBER OF STATIONS UNDER CONSIDERATION                         *
C * NSTR: NUMBER OF STARTING TASKS (CREATED BY THE PROGRAM)            *
C * NTASK: NUMBER OF TASKS                                             *
C * PRED(I,1): NUMBER OF IMMEDIATE PREDECESSORS OF TASK I              *
C * PRED(I,J),J#1: IMMEDIATE PREDECESSORS OF TASK I (NTASK,NTASK)      *
C * S(I): CURRENT PARTIAL SOLUTION (CALC.)                             *
C * SCNT: NUMBER OF VARIABLES IN CURRENT PARTIAL SOLUTION              *
C * SS(I): BEST INCUMBENT SOLUTION (CVAR)                              *
C * SUCC(I,1): NUMBER OF IMMEDIATE SUCCESSORS OF TASK I                *
C * SUCC(I,J),J#1: IMMEDIATE SUCCESSORS OF TASK I (NTASK,MAXSUC)       *
C * TI(I): OPERATION TIME OF TASK I (NTASK)                            *
C * TTASK: NUMBER OF TASKS ENTERED IN CURRENT PARTIAL SOLUTION         *
C * UNLIN(I): ARRAY FOR INDERLINING UNLIN(I)=1 IF I-TH VARIABLE        *
C * IS UNDERLINED, 0 OTHERWISE(CVAR)                                   *
C *-----------------------------------------------------------------------*
C * DATA INPUT: (FORMATTED)                                             *
C * 1-FIRST DATA CARD CONTAINS:                                        *
C * EXPLANATION: VARIABLE: COLUMNS: FORMAT:                             *
C * # OF STATIONS NST 8-10 (I3)                                        *
C * # OF TASKS NTASK 18-20 (I3)                                        *
C * \# OF MAX. SUCCESSORS: MANSUC 28-30 (I3) *
C * SHIFT TIME     ICC  38-40 (I3) *
C * \# OF PRODUCT TYPE    TYPE  58-60 (I3) *
C *
C * 2- AFTER THE FIRST CARD ONE DATA CARD IS REQUIRED FOR *
C * EACH TASK. EACH CARD WILL CONTAIN: *
C *
C * EXPLANATIONS: VARIABLE: COLUMNS: FORMAT: *
C * TASK NUMBER I  1-2 (12) *
C * TASK TIME T(I)  4-5 (12) *
C * GROUP INDEX G(I)  7-8 (12) *
C * \# OF ITS SUCCESSORS SUCC(I,1) 16-17 (12) *
C * ITS SUCCESSORS SUCC(I,2) 18-19, 20-21, ... 24(12) *
C
C******************************************************************************

IMPLICIT INTEGER (A-Z)
REAL CC, EXST, STIME, STMTO, ESIN, EB(20)
COMMON /FTSUC,/MAT(30,125),E(30),NSR,TTASK
COMMON /FTCON/ ARR(3,125),S(125),INLIN(125),CVAR,SCNT
COMMON /FTCON1/ BOLMAT(15,25),NNST
COMMON/FTC/C(125),GMAT(30,30),G(30),FTY,ISTOP,FLN,STIME,ISTART
COMMON /FTSUST/ T(30),NST
COMMON /STBAC/ EST(30),LST(30),INDA(30)
COMMON /FTPOST/ NTASK
DIMENSION A(30,30),SUCC(30,20),SS(125),LOC(30),PRED(30,20)
DIMENSION LAST(20),GROUP(20)

TTASK=0
STIME=0
ISTOP=0
NSR=0
CVAR=0
SLAST=0
NFIN=0
SCNT=0
FLN=0
READ(5,1) NST,NTASK,MANSUC,ICC,TYPE
MAXP2=MANSUC+2.
DO 656 I=1,NTASK
656 READ(5,657) SUCC(I,1),T(I),G(I),(SUCC(I,J),J=2,MAXP2)
NST=NST
FTY=2**TYPE
DO 651 ME=1,NTASK
651 STIME=STIME+T(ME)
CC=ICC
STMTO=STIME
DO 658 I=1,NTASK
SUC1=2*SUCC(I,2)
IF(SUC1.EQ.2) GOTO 658
WRITE(6,658) SUCC(I,1),SUCC(I,J),J=3,SUC1
658 CONTINUE.
C
C FINDING ALL SUCCESSORS OF ALL TASKS AND LATEST STATIONS
C
CALL FOREAC(A, SUCC)
CALL STLIN(0, A, CC)
IF(SUCC(I, 2).EQ.0) GOTO 410
SUCA=SUCC(I, 2)+2
DO 420 J=3, SUCA
IF(IK .NE. SUCC(I, J)) GOTO 420
NLOC=NLOC+1
DO 400 IK=1, NTASK
NLOC=0
DO 410 I=1, NTASK
IF(SUCC(I, 2).EQ.0) GOTO 410
SUCA=SUCC(I, 2)+2
DO 420 J=3, SUCA
IF(IK .NE. SUCC(I, J)) GOTO 420
NLOC=NLOC+1
LOC(NLOC)=I
420 CONTINUE
410 CONTINUE
PRED(IK, 1)=IK
PRED(IK, 2)=NLOC
IF(NLOC.EQ.0) GOTO 400
DO 430 L=1, NLOC
L2=L+2
430 PRED(IK, L2)=LOC(L)
400 CONTINUE
DO 442 I=1, NTASK
DO 443 J=1, NST
443 GMA(I, J)=0
442 CONTINUE
DO 450 I=1, NTASK
KLM=PRED(I, 2)
IF(KLM.EQ.0) GOTO 450
KKLY=KLY+2
WRITE(6, 455) PRED(I, 1), (PRED(I, J), J=3, KKLY)
450 CONTINUE
CALL FORBAC(A, PRED)
CALL STLIN(1, A, CC)

C---------------------- C
C CREATING DECISION VARIABLES C
C---------------------- C
DO 800 IST=1, NST
DO 810 ITS=1, NTASK
IF((IST.LT.LEST(ITS)).OR.(IST.GT.LEST(ITS))) GOTO 810
CVAR=CVAR+1
ARR(1, CVAR)=IST
ARR(2, CVAR)=ITS
ARR(3, CVAR)=0
810 CONTINUE
800 CONTINUE
DO 776 I=1, CVAR
IR2=ARR(1, I)
IR3=ARR(2, I)
776 BGNMAT(IR2, IR3)=I
C---------------------- C
DO 177 NI=1,NTASK
   IF(PRED(I,2).NE.0) GOTO 177
   NSTR=NSTR+1
177 CONTINUE
   DO 178 NI=1,NTASK
   IF(SUCI(I,2).NE.0) GOTO 178
   NFIN=NFIN+1
   SLAST=SLAST+T(I)
   IF(NFIN.EQ.1) LASTMI=I
   IF(NFIN.EQ.1) TEMP1=T(I)
   IF(TEMP1.EQ.T(I)) GOTO 178
   LASTM=I
   TEMP1=T(I)
178 CONTINUE
   NLAST=NTASK-NSTR
   DO 770 NT1=1,NLSTR
   S(TH)=0
   UNLIN(TH)=0
   C(TH)=0
   DO 770 NT2=1,NLAST
   MAT(TG,TH)=0
   770 CONTINUE

C CREATING PRECEDENCE MATRIX C
C

IROW=0
   NT1=NST-1
   DO 820 NT2=1,NTASK
   IF(PRED(ITS,2).EQ.0) GOTO 820
   IROW=IROW+1
   PN=PRED(ITS,2)
   DO 830 ICU=1,NV
   IF(ARR(2,ICV).NE.ITS) GOTO 830
   IQ=ARR(1,ICV)
   TMP=0
   IF(IQ.EQ.NST) GOTO 841
   DO 840 I=IQ,NT1
   TMP=TMP+PN**I
841 TMP=TMP+I
   MAT(IROW,ICV)=-PN*TMP
830 CONTINUE
   PN2=PN+2
   DO 850 IK=3,PN2
   DO 860 ICU=1,NV
   IF(PRED(ITS,IK).NE.ARR(2,ICV)) GOTO 860
   IQ=ARR(1,ICV)
   TMP=0
   IF(IQ.EQ.NST) GOTO 871
   DO 870 I=IQ,NT1
   TMP=TMP+PN**I
871 TMP=TMP+I
   MAT(IROW,ICV)=TMP
CONTINUE
CONTINUE
CONTINUE

CREATING OBJECTIVE FUNCTION

SIMTO = SIMTO / CC
ISIMTO = SIMTO
IF((SIMTO - ISIMTO).EQ.0) GOTO 677
ISIMTO = ISIMTO + 1
677
N01 = ISIMTO + 1
DO 678 JARR = 1, CVAR
C(JARR) = FTY
IF(MARR(1, JARR).LT.N01) GOTO 678
IARR = MARR(2, JARR)
IF(SUCC(IARR, 2).NE.0) GOTO 678
C(JARR) = C(JARR) + FTY * (T(IARR) * ((SLAST + 1)**(MARR(1, JARR) - N01)))
678
CONTINUE
ZBARR = 2**18
DO 100 I = 1, NST
LAST(I) = 0
100
B(I) = CC
NSTP = NST + 1
NSTP1 = NST + NTASK - NST
DO 101 I = NSTP, NSTP1
101
B(I) = 0
DO 179 I = 1, CVAR
IF(LASTMNE.AR(2, I)) GOTO 179
ILOC = AR(1, I)
LAST(ILOC) = I
179
CONTINUE
IC = 1
ISTART = 1

SOLUTION PROCEDURE
FEASIBILITY CHECK AND SAVING THE BETTER FEASIBLE SOLUTION

256 IF(TTASK.LT.NTASK) GOTO 200
C
-------------------------> STEP 6 <-------------------------
C
IF(FUN.GE.ZBARR) GOTO 685
ZBARR = FUN
ISCNT = SCNT
DO 50 IJ = 1, SCNT
50
SS(IJ) = S(IJ)
685
CONTINUE
CALL FATIGUE(SUCC, PRED)
IF(ISTOP.EQ.1) GOTO 99
C
SEARCH OF A NEW VARIABLE TO BE ENTERED IN PARTIAL SOLUTION
C

200 IF(SCNT.GE.CVAR) GOTO 685
DO 103 II=ISTART,CVAR
IF(ARR(3,II).EQ.1) GOTO 103
J=II
II=ARR(1,J)
J1=ARR(2,J)
GOTO 102
103 CONTINUE
WR=12
WRITE(6,9999) WR
102 IF(IT.GT.IO) GOTO 440
C

C ENTERING A NEW VARIABLE IN-PARTIAL SOLUTION
C

C

IO=II
JO=J1
221 GJ1=G(JO)
IF(GJ1.EQ.0) GOTO 222
VAL=CMAT(GJ1,IO)
CTEMP=C(J)/(2**VAL)
GOTO 112
222 CTEMP=C(J)
112 IF((CTEMP+FUN).GT.ZBAR) GOTO 10
IF(PRED(JO,2).EQ.0) GOTO 12
LOG1=B(IO)+T(JO)
IF(LOG1.GT.0) GOTO 10
INTASK=JO-NSTR
INT=INTASK+NST
LOG2=CMAT(INTASK;J)+B(INT)
IF(LOG2.LT.0) GOTO 10
12 SCNT=SCNT+1
S(SCNT)=J
STIME=STIME-T(JO)
FIN=FIN+CTEMP
ARR(3,J)=1
IF(IO.EQ.LST(JO)) UNLIN(SCNT)=1
IF(GJ1.GT.0) CMAT(GJ1,IO)=CMAT(GJ1,IO)+1
CALL SUCCAL(1,JO,SUCC,J,IO,PRED)
CALL COMPL(IO,JG)
ISTART=J+1
GOTO 256
10 SCNT=SCNT+1
S(SCNT)=J
UNLIN(SCNT)=1
ARR(3,J)=1
ISTART=J+1
GOTO 200

C

C FINDING ESTIMATED OBJ. FIN. VALUE FOR CURRENT PARTIAL SOLUTION
C

C

440 EXST=STIME/CC
IEXST=EXST
IF((EXST-IEXST).LE.0) GOTO 402
IEXST-IEXST+1
402  I2=I2+IEXST
IF(I2.GT.NST) GOTO 501
N2=LAST(I2)
ZEST=FIN+CN2
IF(ZEST.LE.ZBARR) GOTO 220
CONTINUE
CALL FATHON(SICC,PRED)
IF(ISTOP.EQ.1) GOTO 99
GOTO 200
220  IO=I1
JO=J1
GOTO 221

C C C C
O P T I M A L  S O L U T I O N  H A S  B E E N  F O U N D
C C C

59 CONTINUE
DO 4 I=1,NTASK
4  WRITE(6,14) I,T(I)
WRITE(6,886)
WRITE(6,11) NST,NTASK,ICC
DO 255 J=1,NST
255  E(J)=ICC
CMAX=0
DO 560 I=1,NTASK
CMAX=MAX(CMAX,G(I))
WRITE(6,886)
WRITE(6,1509)
DO 1450 I=1,CMAX
DO 232 T12=1,9
232  GROUP(T12)=0
GCOUN=0
DO 1460 J=1,NTASK
IF(I.NE.G(J)) GOTO 1460
GCOUN=GCOUN+1
GROUP(GCOUN)=J
1460 CONTINUE
WRITE(6,1510) I,(GROUP(K),K=1,9)
1450 CONTINUE
WRITE(6,1509)
1510 FORMAT(1X,1 GROUP,I3,9I4)
1509 FORMAT(1X,SL,ISCN
I)
IF(ISI.LT.0) GOTO 451
ARR3=ARR2(ISI)
ARR2=ARR1(ISI)
B(ARR2)=B(ARR2)-T(ARR3)
451 CONTINUE
WRITE(6,899)
WRITE(6,1611)
DO 1550 I=1,NST

DO 233 T12=1,9
GROUP(T12)=0
CC0N=0
DO 1560 J=1,ISCNT
ISI=SS(J)
IF(ISI.LT.0) GOTO 1560
IF(ARR(1,ISI).NE.1) GOTO 1560
CC0N=CC0N+1
GROUP(CC0N)=ARR(2,ISI)
1560 CONTINUE
IF(CC0N.EQ.0) GOTO 1570
WRITE(6,1610) I,(GROUP(K),K=1,9),B(I)
ISTEP=I
1570 CONTINUE
WRITE(6,1611)
EMIN=1000
DO 2451 I=1,ISTEP
2451 EMIN=MIN(EMIN,EB(I))
DO 2452 I=1,ISTEP
2452 B(I)=EB(I)-EMIN
BSIN=0
DO 2453 I=1,ISTEP
2453 BSIN=BSIN+B(I)**2
BSIN=SQRT(BSIN)
WRITE(6,2424) BSIN
2424 FORMAT('SMOOTHNESS INDEX=',F5.2)
WRITE(6,886)
1611 FORMAT(1X,70(' '))
1610 FORMAT(1X,'STATION','I3',' ','O14','SLACK TIME','I3','/
STOP
1 FORMAT('/4(7X,I3),17X,I3)
11 FORMAT('/ OF STATION','T23,=','I2,' OF TASKS','T23,=','I2,'
& REQUIRED CYCLE TIME','T23,=','I2)
14 FORMAT('TIME REQUIRED FOR TASK','I3,=','I3
455 FORMAT('IMMED. PRED OF','I2,=','15I4)
657 FORMAT(3(I2,1X),6X,25I2)
659 FORMAT(5X,'IMMED. SUCCESSION OF','I2,=','15I4)
888 FORMAT(15('))
889 FORMAT(1H1)
888 FORMAT('GROUPING REQUIREMENTS :/1X,23("")
895 FORMAT('STATION ASSIGNMENTS :/1X,22("")
9999 FORMAT(10X,"WARNING","I2)
END
C*************************************************************************
C SUBROUTINE FOR FATHOMING CURRENT PARTIAL SOLUTION
C*************************************************************************
SUBROUTINE FATHOM(SUC0,FREE)
IMPLICIT INTEGER (A-Z)
REAL STIME
COMMON /FTSUC/MAT(30,125),E(30),NSTR,TTASK
COMMON /FTCON/, ARR(3,125),S(125),INLIN(125),CVAR,SCNT
COMMON/FT/C(125),CMAT(30,30),C(30),FTY,ISTEP,FUN,STIME,ISTART
COMMON /FTCOM1/ BOUMAT(15,25),NNST
COMMON /FTSUST/, T(30),NST
COMMON /FTFOST/, NTASK
DIMENSION SUCC(30,20),PRED(30,20)
WRL=7
IF(UNLIN(SCNT).EQ.1) GOTO 10
S1=S(SCNT)
S(SCNT)=-S(SCNT)
UNLIN(SCNT)=1
IF(S1)21,22,23
21
SS1=IABS(S1)
L1=ARR(I,SS1)
KT1=ARR(2,SS1)
STIME=STIME+T(KT1)
GJ1=L(KT1)
IF(GJ1.EQ.0) GOTO 222
VAL=CMAT(GJ1,L1)
TEMP=C(SS1)/(2**VAL)
CMAT(GJ1,L1)=CMAT(GJ1,L1)+1
GOTO 223
222
TEMP=C(SS1)
223
FUN=FUN+TEMP
CALL SUCCAL(1,KT1,SUCC,SS1,L1,PRED)
CALL COMPL(L1,KT1)
GOTO 55
23
KT1=ARR(2,S1)
L1=ARR(I,S1)
STIME=STIME+T(KT1)
GJ1=L(KT1)
IF(GJ1.EQ.0) GOTO 323
IF(CMAT(GJ1,L1).EQ.0) WRITE(6,9999) WRL
CMAT(GJ1,L1)=CMAT(GJ1,L1)-1
VAL=CMAT(GJ1,L1)
TEMP=C(S1)/(2**VAL)
GOTO 427
323
TEMP=C(S1)
427
FUN=FUN-TEMP
CALL SUCCAL(-1,KT1,SUCC,S1,L1,PRED)
ISTART=S1+1
GOTO 55
22
WR=S1
WRITE(6,9999) WR
GOTO 55
10
DO 100 IIJ=1,SGAT
I=SCNT+I-IIJ
IF(UNLIN(I).EQ.1) GOTO 100
ITEMP=1
S2=S(I)
S(I)=-S(I)
UNLIN(I)=1
IF(S2)25,22,26
25
SS2=IABS(S2)
L1=ARR(I,SS2)
KT2=ARR(2, SS2)
STIME=STIME+T(KT2)
GJ1=G(KT2)
IF(GJ1.EQ.0) GOTO 333
VAL=GMAT(GJ1, L1)
TEMP=C(SS2)/(2**VAL)
GMAT(GJ1, L1)=GMAT(GJ1, L1)+1
GOTO 334
333 TEMP=C(SS2)
334 FIN=FIN+TEMP
CALL SUCCAL(1, KT2, SUCC, SS2, L1, PRED)
CALL COMPL(L1, KT2)
GOTO 20
26 KT3=ARR(2, S2)
L1=ARR(1, S2)
STIME=STIME+T(KT3)
GJ1=G(KT3)
IF(GJ1.EQ.0) GOTO 444
IF(GMAT(GJ1, L1).EQ.0) WRITE(6, S9999) WRU
GMAT(GJ1, L1)=GMAT(GJ1, L1)-1
VAL=GMAT(GJ1, L1)
TEMP=C(S2)/(2**VAL)
GOTO 200
444 TEMP=C(S2)
200 FIN=FIN-TEMP
CALL SUCCAL(-1, KT3, SUCC, S2, L1, PRED)
ISTART=S2+1
GOTO 20
100 CONTINUE
ISTOP=1
GOTO 55
20 ITEMP=ITEMP+1
DO 101 I=ITEMP, SCNT
S5=S(I)
S(I)=0
UNLIN(I)=0
S55=ABS(S5)
ARR(3, S55)=0
IF(S5.LE.0) GOTO 101
KT5=ARR(2, S5)
L1=ARR(1, S5)
GJ1=G(KT5)
IF(GJ1.EQ.0) GOTO 555
IF(GMAT(GJ1, L1).EQ.0) WRITE(6, S9999) WRU
GMAT(GJ1, L1)=GMAT(GJ1, L1)-1
VAL=GMAT(GJ1, L1)
TEMP=C(S5)/(2**VAL)
GOTO 230
555 TEMP=C(S5)
230 FIN=FIN-TEMP
CALL SUCCAL(-1, KT5, SUCC, S5, L1, PRED)
STIME=STIME+T(KT5)
101 CONTINUE
SCNT=ITEXP
55 RETURN
9999 FORMAT(IOX, 'WARNING=',I2)
END

C******************************************************************************
C
C IF A TASK IO IS ASSIGNED TO A STATION ALL THE OTHER VARIABLES CORRESPONDING TO THIS TASK ARE AUGMENTED TO 0
C
C******************************************************************************

SUBROUTINE COMPL(IO,JO)
IMPLICIT INTEGER (A-Z)
COMMON /FTCOM/ ARR(3,125),S(125),LNLIN(125),CVAR,SCNT
COMMON /STFBAC/ EST(30),LST(30),INDA(30)
COMMON /FTCOM1/ BOMAT(15,25),NINST
IF((IO.EQ.NNST).OR.(IO.EQ.LST(J0))) GOTO 101
IOP1=10+1
LSJO=LST(J0)
DO 100 IIC=IOP1,LSJO
SCNT=SCNT+1
BIJ=BOMAT(IIO,JO)
S(SCNT)=BIJ
LNLIN(SCNT)=1
ARR(3,BIJ)=1
100 CONTINUE
101 CONTINUE
102 RETURN
END

C******************************************************************************
C
C SUBROUTINE SUCCAL DETERMINES THE IMMEDIATE SUCCESSORS OF A TASK. IT ALSO UPDATES THE CORRESPONDING ENTRIES OF RHS VECTOR IF THIS PARTICULAR TASK IS ENTERED IN THE CURRENT PARTIAL SOLUTION AS 1
C
C******************************************************************************

SUBROUTINE SUCCAL(SIGN,J1,SUCC,J,J1,PRED)
IMPLICIT INTEGER (A-Z)
COMMON /FTSIC/ MAT(30,125),B(30),NST,TTASK
COMMON /FTSUST/ T(30),NST
DIMENSION SUCC(30,20),PRED(30,20)
INDSTR=0
IF(SUCC(J1,2).NE.0) GOTO 107
GOTO 55
107 IF(PRED(J1,2).NE.0) GOTO 66
INDSTR=1
66 TT=SUCC(J1,2)+2
DO 109 I=3,TT
UPTASK=SUCC(J1,I)-NST
UPROW=UPTASK+NST
105 B(UPROW)=B(UPROW)+SIGN*MAT(UPTASK,J)
IF(INDSTR.EQ.1) GOTO 77
55 INJ1=J1-NSTR
INDJ2=INDJ1+NST
B(INDJ2)=B(INDJ2)+SIGN*MAT(INDJ1,J)
77 TTASK=TTASK+SIGN
B(I1)=B(I1)+SIGN*T(J1)
RETURN
END

C******************************************************************************C
C SUBROUTINE WHICH DETERMINES ALL SUCCESSORS AND PREDECESSORS OF A PARTICULAR TASK
C******************************************************************************C
SUBROUTINE FORBAC(A,SUCC)
IMPLICIT INTEGER(A-Z)
COMMON /STFBAC/ EST(30),LST(30),INDA(30)
COMMON /PTPOST/ N TASK
DIMENSION SUCC(30,20),A(30,30),INDEX1(30),INDEX2(30)
DO 100 IJ=1,NTASK
I=IJ
NSUC=1
A(IJ,NSUC)=I
IF(SUCC(IJ,2).EQ.0) GOTO 101
IND2=SUCC(IJ,2)
DO 300 IT=1,IND2
IT2=IT+2
INDEX2(IT)=SUCC(IJ,IT2)
NSUC=NSUC+1
A(IJ,NSUC)=SUCC(IJ,IT2)
300 CONTINUE
201 IND1=0
DO 687 IK=1,IND2
I=INDEX2(IK)
C IF((SUCC(IJ,2).EQ.0).AND.(IK.EQ.IND2)) GOTO 101
IF(SUCC(IJ,2).EQ.0) GOTO 687
SUCC2=SUCC(IJ,2)
DO 30 K=3,SUCC1
II=SUCC(IJ,K)
DO 40 KK=1,NSUC
IF(II.EQ.A(IJ,KK)) GOTO 30
40 CONTINUE
NSUC=NSUC+1
IND1=IND1+1
INDEX1(IND1)=II
A(IJ,NSUC)=II
30 CONTINUE
687 CONTINUE
IF(INEL.EQ.0) GOTO 101
IND2=IND
DO 150 M=1,IND2
150 INDEX2(M)=INDEX1(M)
GOTO 201
101 INDA(IJ)=NSUC
100 CONTINUE
RETURN
END
SUBROUTINE STLM(INEST,A,CC)
IMPLICIT INTEGER (A-Z)
REAL CC,SMT
COMMON /FTSUBT/ T(30),NST
COMMON /STFBAC/ EST(30),LST(30),INDA(30)
COMMON /FTFOST/ NTASK
DIMENSION A(30,30)
DO 600 I=1,NTASK
SMT=0
ENDA=INDA(I)
DO 610 IR=1,ENDA
IIR=A(I,IR)
610 SMT=SMT+I(IIR)
SMT=SMT/CC
ISMT=SMT
IF((SMT-ISMT).LE.0) GOTO 601
ISMT=ISMT+1
601 EST(I)=ISMT
600 CONTINUE
IF(INEST.EQ.1) GOTO 666
DO 620 IR=1,NTASK
620 LST(IR)=NST+1-EST(IR)
666 RETURN
END
APPENDIX C2

COMPUTER PROGRAM OF THE ALGORITHM DESCRIBED IN CHAPTER 5
C * ************************************************************
C * VARIABLES USED IN THE PROGRAM:
C * (THE VARIABLES GIVEN IN PARANATHESSES INDICATE THE MINIMUM DIMENSIONS
C * REQUIRED FOR THE ARRAYS)
C *
C * A(I,J): ARRAY WHICH CONTAINS ALL PREDECESSORS/SUCCESSIONS
C * OF TASK I (NTASK,NTASK)
C * ARR(I,J) WORKING ARRAY (J,K) (NVAR)
C * B(N): RHS VECTOR (CREATE BY THE PROGRAM) (NTASK)
C * BOUND(I,J): WORKING VECTOR (NTASK)
C * C(I): OBJECTIVE FUNCTION COEFFICIENTS (CREATE BY THE
C * PROGRAM)
C * CVAR: TOTAL NUMBER OF INDEPENDENT VARIABLES GENERATED BY
C * THE PROGRAM. CVAR DEPENDS ON THE NVAR, NTASK, TASK
C * TIMES, TYPE OF NETWORK, A MAXIMUM VALUE THAT CVAR
C * MAY TAKE IS (NTASK*NSTR)
C * E(COST(I): DUPLICATION COST OF TASK I (NTASK)
C * EST(I): EARLIEST STATION WHERE TASK I CAN BE ASSIGNED (NTASK)
C * F(I): CURRENT OBJECTIVE FUNCTION VALUE
C * G(I): GROUP TO WHICH TASK I BELONGS TO (NTASK)
C * GMAT(I,J): WORKING ARRAY (NTASK,NTASK)
C * GROUP(I): WORKING ARRAY FOR PRINTING RESULTS (1C)
C * ICC: SHIFETIME
C * IND(M): WORKING ARRAY (NTASK)
C * LAST(I) WORKING ARRAY (NTASK)
C * LOG(I) WORKING ARRAY (NTASK)
C * LST(I): LATEST STATION WHERE TASK I CAN BE ASSIGNED (NTASK)
C * MAT(I,J): PRECEDENCE MATRIX (CREATE BY THE PROGRAM)
C * MAXSUC: NUMBER OF MAXIMUM IMMEDIATE SUCCESSOR
C * NST: NUMBER OF STATIONS UNDER CONSIDERATION
C * NSTR: NUMBER OF STARTING TASKS (CREATE BY THE PROGRAM)
C * NTASK: NUMBER OF TASKS
C * PRED(I,1): NUMBER OF IMMEDIATE PREDECESSORS OF TASK I
C * PRED(I,J),#1: IMMEDIATE PREDECESSORS OF TASK I (NTASK,NTASK)
C * S(I): CURRENT PARTIAL SOLUTION (CVAR)
C * SCNT: NUMBER OF VARIABLES IN CURRENT PARTIAL SOLUTION
C * SS(I): BEST INCUMBENT SOLUTION (CVAR)
C * SCONT: COST OF OPENING A NEW STATION
C * SUC(I,1): NUMBER OF IMMEDIATE SUCCESSORS OF TASK I
C * SUC(I,J),#1: IMMEDIATE SUCCESSORS OF TASK I (NTASK,MAXSUC)
C * T(I): OPERATION TIME OF TASK I (NTASK)
C * TTASK: NUMBER OF TASKS ENTERED IN CURRENT PARTIAL SOLUTION
C * UNLIN(I): ARRAY FOR UNDERLINED UNLIN(I)=1 IF I TH VARABLE
C * IS UNDERLINED, 0 OTHERWISE (CVAR)
C *
C * ************************************************************
C *
C * DATA INPUT: (FORMATTED)
C *
C * 1-FRST DATA CARD CONTAINS:
C *
C * EXPLANATION: VARIABLE: COLUMNS: FORMAT:
C * & OF STATIONS
C * & OF TASKS
C * & OF MAX. SUCCESSORS:
C * SHIFT TIME
C * COST OF NEW STATION
C * & OF PRODUCT TYPE
C *
C * 2. AFTER THE FIRST CARD ONE DATA CARD IS REQUIRED FOR
C * EACH TASK. EACH CARD WILL CONTAIN:
C *
C * EXPLANATIONS:
C * TASK NUMBER
C * TASK TIME
C * GROUP INDEX
C * DUPLICATION COST
C * & OF IPS SUCCESSORS
C * ITS SUCCESSORS
C
C*******************************************************************************

IMPLICIT INTEGER (A-Z)
REAL CC, EST, STIME, STIMO, BSTY, EB(20)
COMMON /FTSC(1): NST(30,150), S(30), NSTR, TTASK
COMMON /FTCON/: ARR(3,150), S(150), CNLITK(150), CVAR, SCNT
COMMON /FTCON/: BOUND(15,25), NNST
COMMON /FT/C(150), G(30), ISTOP, FIN, STIME, ISTART
COMMON /FTD/E/: CHAT(30,30), ECOST(30), STP1, GMAX
COMMON /FTD/E/: T(25), NST
COMMON /FTBAC/: EST(25), LST(25), INDA(30)
COMMON /FTFOST/: NTASK
DIMENSION L(30,30), SUCC(25,20), SS(150), LOC(36), PRED(25,20)
DIMENSION LAST(20), GROUP(20)
READ(5,1) NST, NTASK, MAXSUC, ICC, ECOST
MAXP = MAXSUC + 2
DO 656 I = 1, NTASK

656 READ(5,657) SUCC(I,1), T(I), G(I), ECOST(I), (SUCC(I,J), J = 2, MAXP)
CALL TINIT
TTASK = 0
STIME = 0
ISTOP = 0
NSTR = 0
CVAR = 0
SCNT = 0
FIN = 0
CC = ICC
NST = NST
STP1 = NST + 1
GMAX = 0
DO 560 I = 1, NTASK

560 GMAX = MAX0(C(I), GMAX)
DO 651 ME = 1, NTASK

651 STIME = STIME + T(ME)
SIMD = STIME

C
C FINDING ALL SUCCESSORS OF ALL TASKS AND LATEST STATIONS
C
CALL FORBAC(A,SUCC)
CALL STLIM(0,A,CC)

CREATING ALL PREDECESSORS OF ALL TASKS AND EARLIEST STATIONS

DO 400 IK=1,NTASK
   NLOC=0
   DO 410 I=1,NTASK
      IF(SUCC(I,2).EQ.0) GOTO 410
      SUCA=SUCC(I,2)+2
      DO 420 J=3,SUCA
      IF(IK.NE.SUCC(I,J)) GOTO 420
      NLOC=NLOC+1
      LOC(NLOC)=I
   420 CONTINUE
   410 CONTINUE
   PRED(IK,1)=IK
   PRED(IK,2)=NLOC
   IF(NLOC.EQ.0) GOTO 400
   DO 430 L=1,NLOC
      L2=L+2
   430 PRE(IK,L2)=LOC(L)
   400 CONTINUE
   DO 442 I=1,NTASK
      DO 442 J=1,STP1
      CVAL(I,J)=0
      CALL FORBAC(A,PRED)
      CALL STLIM(1,A,CC)

CREATING DECISION VARIABLES

DO 800 IST=1,NST
   DO 810 ITS=1,NTASK
      IF((IST.LT.EST(ITS)).OR.(IST.GT.LST(ITS))) GOTO 810
      CVAR=CVAR+1
      ARR(1,CVAR)=IST
      ARR(2,CVAR)=ITS
      ARR(3,CVAR)=0
   810 CONTINUE
   800 CONTINUE
   DO 776 I=1,CVAR
      IR2=ARR(1,I)
      IR3=ARR(2,I)
   776 CVAL(IR2,IR3)=1

COUNTING STARTING AND FINAL TASKS

DO 177 I=1,NTASK
   IF(PRED(I,2).NE.0) GOTO 177
   NSIR=NSIR+1
177 CONTINUE
   DO 178 I=1,NTASK
IF(SUC(i,2).NE.0) GOTO 178
LASTM=I
GOTO 321
178 CONTINUE
321 NLAST=NTASK-NSTR
   DO 770 TH=1,CVAR
      S(TH)=0
      LINL(TH)=0
      C(TH)=0
      DO 770 TG=1,NLAST
    770 MAT(TG,TH)=0
C  CREATING PRECEDENCE MATRIX  C
C
   IROW=0
   NT1=NST-1
   DO 820 ITS=1,NTASK
      IF(PRED(ITS,2).EQ.0) GOTO 820
      IROW=IROW+1
      PN=PRED(ITS,2)
      DO 830 ICV=1,CVAR
         IF(ARR(2,ICV).NE.ITS) GOTO 830
         ICV=ARR(1,ICV)
         TMP=0
         IF(IQ.EQ.NST) GOTO 841
         DO 840 I=IQ,NT1
            TMP=TMP+PN**I
          840 CONTINUE
          TMP=TMP+1
          MAT(IROW,ICV)=-PN*TMP
        830 CONTINUE
      841 MAT(IROW,ICV)=TMP
   830 CONTINUE
   PK2=PN+2
   DO 850 IK=3,PK2
      DO 860 ICV=1,CVAR
         IF(PRED(IK,ICV).NE.ARR(2,ICV)) GOTO 860
         ICV=ARR(1,ICV)
         TMP=0
         IF(IQ.EQ.NST) GOTO 871
         DO 870 I=IQ,NT1
            TMP=TMP+PN**I
          870 CONTINUE
          TMP=TMP+1
          MAT(IROW,ICV)=TMP
        860 CONTINUE
      850 CONTINUE
      820 CONTINUE
C  CREATING OBJECTIVE FUNCTION  C
C
   SUMP=SUMP/CC
   ISUMP=ISUMP
   IF((SUMP-ISUMP)**2.EQ.0) GOTO 677
   ISUMP=ISUMP+1
   677 DO 678 JARR=1,CVAR
      C(JARR)=0
IF(ARR(1,JARR).LE.ISLMTO) GOTO 678
JARR=ARR(2,JARR)
IF(SUC(IARR,2).NE.0) GOTO 678
C(JARR)=(ARR(1,JARR)-ISLMTO)*STCOST
678 CONTINUE
ZBAR=Z**18
DO 100 I=1,NST
LAST(I)=0
100 E(I)=-CC
NSTR=NST+1
NSTP1=NST+NTASK-NSTR
DO 101 I=NSTP,NSTP1
101 B(I)=0
DO 175 I=1,CVAR
IF(LAST(I).NE.ARR(2,I)) GOTO 175
ILOC=ARR(1,I)
LAST(ILOC)=I
175 CONTINUE
IO=1
ISTART=1
CALL TUSED(TIME1)
CALL TINIT

C
C SOLUTION PROCEDURE
C
C FEASIBILITY CHECK AND SAVING THE BETTER FEASIBLE SOLUTION
C

256 IF(TTASK.LT.NTASK) GOTO 200
IF(FIN.GE.ZBAR) GOTO 685
ZBAR=FIN
ISCNT=SCNT
DO 50 IJ=1,SCNT
50 SS(IJ)=S(IJ)
685 CALL FATHOM(SUC,PRED)
IF(ISTOP.EQ.1) GOTO 99

C
C SEARCH OF A NEW VARIABLE TO BE ENTERED IN PARTIAL SOLUTION
C

200 IF(SCNT.GE.CVAR) GOTO 685
DO 103 II=ISTART,CVAR
IF(ARR(3,II).EQ.1) GOTO 103
J=II
II=ARR(1,J)
J1=ARR(2,J)
GOTO 102
103 CONTINUE
WR=12
102 IF(I1.GT.IO) GOTO 440

C ENTERING A NEW VARIABLE IN PARTIAL SOLUTION

C IO=II
JO=J1
221 CJ1=C(J0)
   IF(GJ1.EQ.C) GOTO 222
   CTEMP=DUPCOS(GJ1,I0,J0)+C(J)
   GOTO 112
222 CTEMP=C(J)
112 IF((CTEMP+FUN).CT.2BAR) GOTO 10
   IF(PRED(J0,2).EQ.0) GOTO 12
   LOG1=B(I0)+T(J0)
   IF(LOG1.CT.0) GOTO 10
   INTASK=J0-KSTR
   INT=INTASK+NST
   LOC2=MAT(INTASK,J)+E(INT)
   IF(LOG2.LT.0) GOTO 10
   SCNT=SCNT+1
   S(SCNT)=J
   STIME=STIME+T(J0)
   FIN=FUN+CTEMP
   ARR(3,J)=1
   IF(GJ1.LE.0) GOTO 131
   CALL UPGMAT(GJ1,I0,J0,1)
131 CALL SUCCAL(1,JC,SUCC,J,I0,PRED)
   CALL COMPL(I0,JC)
   ISTART=J+1
   GOTO 256
10 .SCNT=SCNT+1
   S(SCNT)=J
   UNLIN(SCNT)=1
   ARR(3,J)=1
   ISTART=J+1
   GOTO 200

C-----------------------------------------------C
C FINDING ESTIMATED OBJ. FIN. VALUE FOR CURRENT PARTIAL SOLUTION. C
C-----------------------------------------------C
440 EXST=STIME/CC
   IEXT=EXST
   IF((EXST-IEXT).LE.0) GOTO 402
   IEXT=IEXT+1
402 I2=I0+IEXT
   IF(I2.GT.NST) GOTO 501
   IM=LAST(I2)
   ZEST=FIN+C(IM)
   IF(ZEST.LE.2BAR) GOTO 220
501 CONTINUE
   CALL FATHOX(SUCC,PRED)
   IF(ISTOP.EQ.1) GOTO 99
   GOTO 200
220 IO=I1
   JO=J1
   GOTO 221

C-----------------------------------------------C
C OPTIMAL SOLUTION HAS BEEN FOUND                    C
C-----------------------------------------------C
CONTINUE
CALL TUSED(TIME2)
WRITE(6,911) NST,NTASK,ICC,ISUNTO
WRITE(6,1510) STCOST
DO 255 J=1,NST
255 B(J)=ICC
WRITE(6,898)
WRITE(6,1509)
DO 1450 I=1,NMAX
DO 232 T12=1,9
232 GROUP(T12)=0
GCOUNT=0
DO 1460 J=1,NTASK
IF(I.NE.G(J)) GOTO 1460
JJJ=J
GCOUNT=GCOUNT+1
GROUP(GCOUNT)=J
1460 CONTINUE
WRITE(6,1510) I,(GROUP(K),K=1,9),EQCOST(JJJ)
1450 CONTINUE
WRITE(6,1509)
DO 451 I=1,ISCNT
ISI=SS(I)
IF(ISI.LT.0) GOTO 451
ARR3=ARR(2,ISI)
ARR2=ARR(1,ISI)
E(ARR2)=E(ARR2)-T(ARR3)
451 CONTINUE
WRITE(6,899)
WRITE(6,1611)
DO 1550 I=1,NST
DO 233 T12=1,9
233 GROUP(T12)=0
GCOUNT=0
DO 1560 J=1,ISCNT
ISI=SS(J)
IF(ISI.LT.0) GOTO 1560
IF(ARR(1,ISI).NE.1) GOTO 1560
GCOUNT=GCOUNT+1
GROUP(GCOUNT)=ARR(2,ISI)
1560 CONTINUE
IF(GCOUNT.EQ.0) GOTO 1570
WRITE(6,1610) I,(GROUP(K),K=1,9),E(I)
ISTEP=I
1550 CONTINUE
1570 WRITE(6,1611)
ENIN=1000
DO 2451 I=1,ISTEP
2451 BMIN=MINC(ENIN,E(I))
DO 2452 I=1,ISTEP
2452 E(I)=E(I)-ENIN
BSUM=0
DO 2453 I=1,ISTEP
BSLIM=BSLM+BB(I)**2
BSLM=SQR(BSLIM)
WRITE(6,2424) BSLIM
2424 FORMAT(//'SMOOTHNESS INDEX='','E5.2)
WRITE(6,343) TIME1,TIME2,CVAR
343 FORMAT(///,'TIME 1','E17.17',' TIME 2','E17.17',' CVAR','E14.14')
STOP
1 FORMAT((6(7X,I3)))
11 FORMAT(//' # OF STATION',T27,'=','I3',' # OF TASKS',T27,'=','I3','
& ' REQUIRED CYCLE TIME',T27,'=','I3',' MIN. C OF STATION',T27,'=','I3')
151 FORMAT(//' COST OF NEW STATION',T27,'=','I3')
657 FORMAT(3(I2,1X),I4,2X,25I2)
888 FORMAT(///)
889 FORMAT(///,' GROUPING REQUIREMENTS :','/1X,23(*)'))
1509 FORMAT(1X,6E(1'-'))
1510 FORMAT(1X,1' GROUP',I3,1' | ',E14,1' | ',D1P. COST=',E13,1')
1610 FORMAT(1X,1' STATION',I3,1' | ',E14,1' | ',SLACK TIME=',E13,1')
1611 FORMAT(1X,70(1'-'))
END

C***********************************************************************
C
SUBROUTINE TO FATHOM CURRENT PARTIAL SOLUTION
C***********************************************************************
C
SUBROUTINE 'FATHOM(SUCC,PRED)
IMPLICIT INTEGER (A-Z)
REAL TIME
COMMON /FTSUC/NAT(30,150),S(30),KSTR,TTASK
COMMON /FTCOM/ ARR(3,150),S(150),LINL(150),CVAR,SCNT
COMMON /FTT/C(150),C(30),ISTOP,FLN,STIME,START
COMMON /FTDB/ GMAT(30,30),ECOST(30),STP1,CMAT
COMMON /FTCOM/ EQUALMAT(15,25),NST
COMMON /FTSUST/ T(25),NST
COMMON /FTPOST/ NTASK
DIMENSION SUCC(25,20),PRED(25,20)

WRITE
IF(LINL(150),150,T) GOTO 10
S1=S(150)
150 S(150)=S(150)
LINL(150)=1
IF(S1)21,22,23
21 S1=ABS(S1)
L1=ARR(1,S1)
K1=ARR(2,S1)
STIME=STIME-T(KT1)
1G1=C(KT1)
IF(G1.EQ.C) GOTO 222
TEMP=ECOST(G1,L1,KT1)+C(S1)
222 TEMP=C(S1)
1G0=FU+TEMP
CALL SUCCAL(1,KT1,SUCC,SS1,LI,PRED)
CALL COMPL(L1,KT1)
GOTO 55
23 KT1=ARR(2,S1)
L1=ARR(1,S1)
STIME=STIME+T(KT1)
GJ1=G(KT1)
IF(GJ1.EQ.0) GOTO 323
CALL PECMAT(GJ1,L1,KT1,-1)
TEMP=DBPCCS(GJ1,L1,KT1)+C(S1)
GOTO 427
323 TEMP=C(S1)
427 FIN=FIN-TEMP
CALL SUCCAL(-1,KT1,SUCC,S1,L1,PRED)
ISTART=S1+1
GOTO 55
22 WR=5
GOTO 55
10 DO 100 IIJ=1,SCNT
I=SCNT+1-IIJ
IF(UNLIN(I).EQ.1) GOTO 100
ITEMP=I
S2=S(I)
S(I)=-S(I)
UNLIN(I)=1
IF(S2)25,22,26
25 SS2=ABS(S2)
L1=ARR(1,SS2)
KT2=ARR(2,SS2)
STIME=STIME-T(KT2)
GJ1=G(KT2)
IF(GJ1.EQ.0) GOTO 333
TEMP=DBPCCS(GJ1,L1,KT2)+C(SS2)
CALL UPGMAT(GJ1,L1,KT2,1)
GOTO 334
333 TEMP=C(SS2)
334 FIN=FIN+TEMP
CALL SUCCAL(1,KT2,SUCC,SS2,L1,PRED)
CALL COMPL(L1,KT2)
GOTO 20
26 KT3=ARR(2,S2)
L1=ARR(1,S2)
STIME=STIME+T(KT3)
GJ1=G(KT3)
IF(GJ1.EQ.0) GOTO .444
CALL UPGMAT(GJ1,L1,KT3,-1)
TEMP=DBPCCS(GJ1,L1,KT3)+C(S2)
GOTO 200
444 TEMP=C(S2)
200 FIN=FIN-TEMP
CALL SUCCAL(-1,KT3,SUCC,S2,L1,PRED)
ISTART=S2+1
GOTO 20
100 CONTINUE
ISTOP=1
GOTO 55

20  ITEML=ITEMP+1
DO 101 I=ITEML,SCNT
S5=S(I)
S(I)=0
UNLIN(I)=0
S5=IABS(S5)
ARR(3,S5)=0
IF(S5.LE.0) GOTO 101
KT5=ARR(2, S5)
L1=ARR(1, S5)
GJ1=G(KT5)
IF(GJ1.EQ.0) GOTO 55
CALL UPGMAT(GJ1, L1, KT5, -1)
TEMP=SUCOS(GJ1, L1, KT5)+C(S5)
GOTO 230

55  TEMP=C(S5)
230  FIN=FIN-TEMP
CALL SUCCAL(-1, KT5, SUC, S5, L1, PRED)
STIME=STIME+T(KT5)

101  CONTINUE
      SCNT=ITEMP

55  RETURN
END

******************************************************************************C
******************************************************************************C
C                 IF A TASK IO IS ASSIGNED TO A STATION ALL THE OTHER  C
C                 VARIABLES CORRESPONDING TO THIS TASK ARE AUGMENTED TO 1    C
******************************************************************************C
C
SUBROUTINE COMPL(IO, JO)
IMPLICIT INTEGER (A-Z)
COMMON /FCON/ ARR(3,150), S(150), INLIN(150), CVAR, SCNT
COMMON /STFCAG/ EST(25), LST(25), INDA(30)
COMMON /FCON/ EGCMAT(15,25), NSST
IF((IO.EQ.NSST).OR.(IO.EQ.LST(JO))) GOTO 101
IO1=IO+1
LSJO=LST(JO)
DO 100 II0=10P1, LSJO
SCNT=SCNT+1
BIJ=EGCMAT(II0, JO)
S(SCNT)=BIJ
INLIN(SCNT)=1
ARR(3, BIJ)=1

100  CONTINUE

101  RETURN
END

******************************************************************************C
******************************************************************************C
C                 SUBROUTINE SUCCAL DETERMINES THE IMMEDIATE SUCCESSORS OF    C
C                 A TASK. IT ALSO UPDATES THE CORRESPONDING ENTRIES OF RHS   C
C                 VECTOR IF THIS PARTICULAR TASK IS ENTERED IN THE CURRENT    C
C                 PARTIAL SOLUTION AS 1                                       C
******************************************************************************C
SUBROUTINE SUCGAL(SIGN,J1,SUCG,J,II,PRED)
IMPLICIT INTEGER (A-Z)
COMMON /FTSUC/HAT(30,150),B(30),NSTR,TTASK
COMMON /FTSUST/ T(35),NST
DIMENSION SUCG(25,20),PRED(25,20)

INDSR=0
IF(SUCG(J1,2).NE.0) GOTO 107
GOTO 55

107 IF(PRED(J1,2).NE.0) GOTO 66
INDSR=1

66 TT=SUCG(J1,2)+2
DO 106 I=3,TT
UPTASK=SUCG(J1,I)-NSTR
UPROW=UPTASK+NST

109 B(UPROW)=B(UPROW)+SIGN*NAT(UPTASK,J)
IF(INDSR.EQ.1) GOTO 77

55 INDJ1=J1-NSTR
INDJ2=INDJ1+NST
B(INDJ2)=B(INDJ2)+SIGN*NAT(INDJ1,J)

77 TTASK=TTASK+SIGN
B(II)=B(II)+SIGN*T(J1)
RETURN
END

C******************************************************************************************************C
C
C SUBROUTINE WHICH DETERMINES ALL SUCCESSORS AND PREDECESSORS OF A PARTICULAR TASK C
C
C SUBROUTINE FOREAC(A,SUC)
C IMPLICIT INTEGER(A-Z)
C COMMON /STFAC/ EST(25),LST(25),INDA(30)
C COMMON /FTFOST/ NTASK
C DIMENSION SUCG(25,20),A(30,30),INDEX1(30),INDEX2(30)
C DO 100 IJ=1,NTASK
I=IJ
NSUC=1
A(IJ,NSUC)=I
IF(SUCG(I,2).EQ.0) GOTO 101
IND2=SUCG(I,2)
DO 300 IT=1,IND2
IT2=IT+2
INDEX2(IT)=SUCG(I,IT2)
NSUC=NSUC+1
A(IJ,NSUC)=SUCG(I,IT2)

300 CONTINUE

201 IND1=0
DO 667 IK=1,IND2
I=INDEX2(IK)
C IF((SUCG(I,2).EQ.0) .AND. (IK.EQ.INDE2)) GOTO 101
IF(SUCG(I,2).EQ.0) GOTO 667
SUC1=2+SUCG(I,2)
DO 30 K=3,SUC1

II=SUC(I,K)
DO 40 KK=1,NSUC
 IF(II.NE.(IJ,KK)) GOTO 30
40 CONTINUE
NSUC=NSUC+1
IND1=IND1+1
INDEX1(IND1)=II
A(IJ,NSUC)=II
30 CONTINUE
667 CONTINUE
 IF(IND1.EQ.6) GOTO 101
INT2=IND1
 DO 150 M=1,IND2
150 INDEX2(N)=INDEX1(N)
GOTO 201
101 INDA(IJ)=NSUC
100 CONTINUE
RETURN
END

*******************************************************************************
C
*******************************************************************************
C SUBROUTINE STLIN FINDS THE Earliest AND LATEST STATIONS OF A TASK
C
*******************************************************************************
C SUBROUTINE STLIN(INEST,A,CC)
INPLICIT INTEGER (A-Z)
REAL CC,ST
COMMON /FTSUB/ T(25),NST
COMMON /STFAC/ EST(25),LST(25),INDA(3C)
COMMON /FTFST/ NTASK
DIMENSION A(30,30)
DO 600 I=1,NTASK
SMT=0
 ENDA=INDA(I)
 DO 610 IR=1,ENDA
610 SMT=SMT+T(IR)
 SMT=SMT/CC
 ISMT=SMT
 IF((SMT-ISMT).LE.0) GOTO 601
 ISMT=ISMT+1
601 EST(I)=ISMT
600 CONTINUE
 IF(INEST.EQ.1) GOTO 666
 DO 620 IK=1,NTASK
 LST(IK)=NST+1-EST(IK)
 IF(EST(IK).EQ.0) LST(IK)=LST(IK)-1
620 CONTINUE
666 RETURN
END

*******************************************************************************
C
*******************************************************************************
C FUNCTION DUCOS DETERMINES THE DUPLICATION COSTS OF TASKS
C
WHICH BELONG TO A GROUP

FUNCTION DUPCOS(GJ1,IO,JO)
IMPLICIT INTEGER (A-Z)
COMMON /FDTDB/ GMAT(30,30),ECCOST(30),STPL,GMAX
IF(GMAT(GJ1,IO).NE.0) GOTO 30
IF(GMAT(GJ1,STPL)) 20,30,40
20 WRITE(6,10)
10 FORMAT(' ERROR IN DUPCOS'.)
RETURN
DUPCOS=0
RETURN
40 DUPCOS=ECCOST(JO)
RETURN
END

***********************************************************************

SUBROUTINE UPGMAT UPDATES THE SETS Gsy(i), y=1,2,...,GT:

i=1,2,...N (REFERING TO SECTION 5.2)

SUBROUTINE UPGMAT(GJ1,IO,JO,ISIG)
IMPLICIT INTEGER (A-Z)
COMMON /FDTDB/ GMAT(30,30),ECCOST(30),STPL,GMAX
GMAT(GJ1,IO)=GMAT(GJ1,IO)+ISIG
GMAT(GJ1,STPL)=GMAT(GJ1,STPL)+ISIG
RETURN
END
APPENDIX D

SAMPLE PROBLEMS
Figure D.2

precedence diagram of test problem 2
precedence diagram of test problem 4

Figure D.4
REFERENCES


19- Moodie, C.L. and Nardeville, D.E., "Project Resource Balancing


VITA AUCTORIS

1955  Born in Ankara, Turkey, on July 23.

1974  Completed secondary education at the Galatasaray Lisesi, Istanbul, Turkey.

1979  Graduated with Bachelor of Science degree in Industrial Engineering from Bogazici University, Istanbul, Turkey.

1979  Joined University of Windsor for graduate studies.

1981  Currently a candidate for the degree of Master of Applied Science in Industrial Engineering.