INFORMATION TO USERS

This was produced from a copy of a document sent to us for microfilming. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help you understand markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure you of complete continuity.

2. When an image on the film is obliterated with a round black mark it is an indication that the film inspector noticed either blurred copy because of movement during exposure, or duplicate copy. Unless we meant to delete copyrighted materials that should not have been filmed, you will find a good image of the page in the adjacent frame. If copyrighted materials were deleted you will find a target note listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed the photographer has followed a definite method in "sectioning" the material. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For any illustrations that cannot be reproduced satisfactorily by xerography, photographic prints can be purchased at additional cost and tipped into your xerographic copy. Requests can be made to our Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases we have filmed the best available copy.
Ahn, Seokyoo

DEVELOPMENT OF AN ALGORITHM FOR A MULTI-PROCESSOR WEIGHTED FLOW TIME SCHEDULING PROBLEM

The Ohio State University

University Microfilms International

Copyright 1982 by Ahn, Seokyoo

All Rights Reserved
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs or pages
2. Colored illustrations, paper or print
3. Photographs with dark background
4. Illustrations are poor copy
5. Pages with black marks, not original copy
6. Print shows through as there is text on both sides of page
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements
9. Tightly bound copy with print lost in spine
10. Computer printout pages with indistinct print ✓
11. Page(s) ________ lacking when material received, and not available from school or author.
12. Page(s) ________ seem to be missing in numbering only as text follows.
13. Two pages numbered __________. Text follows.
14. Curling and wrinkled pages
15. Other

University Microfilms International
Development of an Algorithm for A
Multi-Processor Weighted Flow
Time Scheduling Problem

DISSERTATION

Presented in Partial Fulfillment of the
Requirement for the Degree Doctor of Philosophy
in the Graduate School of the Ohio State University

by

Seokyoo Ahn, B.S., M.S.

* * * * * *

The Ohio State University
1982

Reading Committee:

Dr. Albert B. Bishop
Dr. Clark A. Mount-Campbell
Dr. Subhash C. Sarin
Dr. Wilbert E. Wilhelm

Approved by:

Albert B. Bishop
Advisor
Department of Industrial and Systems Engineering
ACKNOWLEDGEMENTS

I wish to acknowledge the Department of Industrial and Systems Engineering for providing the environment in which this dissertation was completed.

I would like to express my sincere gratitude to my advisor, Professor Albert B. Bishop, for his assistance and suggestions during the course of this research, and for the most valuable comments on the manuscript of the dissertation.

I would like to express my sincere thanks to Professor Subhash C. Sarin, who provided guidance, advice and many helpful suggestions during the preparation of this dissertation.

A special acknowledgement is due for Professor Clark A. Mount-Campbell for his support and encouragement ever since my arrival at The Ohio State University. I also thank Professor Wilbert E. Wilhelm for serving on my reading committee.

Finally, my deepest appreciation is reserved for my parents for their inspiration and support throughout this educational interlude.
VITA

August 7, 1949 . . . . Born - Seoul, Korea

1973 . . . . . . . . B.S., Physics, The Seoul National University, Seoul, Korea


1978-1980 . . . . Graduate Research Associate, Industrial and Systems Engineering, The Ohio State University, Columbus, Ohio

1980 . . . . . . . . M.S., Industrial and Systems Engineering, The Ohio State University, Columbus, Ohio

1980-1982 . . . . Graduate Teaching Associate, Industrial and Systems Engineering, The Ohio State University, Columbus, Ohio

PUBLICATIONS


FIELD OF STUDY

Major Field: Industrial and Systems Engineering

Scheduling
Mathematical Programming
Operations Research
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>LITERATURE REVIEW</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.1 Survey of Scheduling Problems</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.2 Branch and Bound</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>2.3 Heuristics</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>2.4 Mathematical Programming</td>
<td>12</td>
</tr>
<tr>
<td>III</td>
<td>PROBLEM FORMULATION</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.1 General Notation</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.2 Problem Definition</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>3.3 Job Reordering</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>3.4 Formulation 1</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>3.5 Local Optimum of Formulation 1</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>3.6 Formulation 2</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>3.7 Formulation 3</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>3.8 Formulation 4</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>3.9 Formulation 5</td>
<td>31</td>
</tr>
<tr>
<td>IV</td>
<td>PROPOSED METHODOLOGY: BRANCH AND BOUND METHOD</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>4.1 Fathoming Rules</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>4.1.1 Precedence Matrix</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>4.1.2 Lower Bound-1</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>4.1.3 Lower Bound-2</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>4.1.4 Earliest Starting Time</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>4.2 Branching Rules</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>4.2.1 Branch Rule-1</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>4.2.2 Branch Rule-2</td>
<td>51</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.3 Upperbound and Brute-Force Calculation of Computer Time</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>V EXPERIMENTATION AND PROGRAM DEVELOPMENT</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>5.1 Dynamic Core Allocation</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>5.2 Problem Generation</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>5.3 Program Development</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>VI RESULTS, CONCLUSIONS AND RECOMMENDATION</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>6.1 Results and Conclusions</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>6.2 Recommendations for Future Research</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>APPENDICES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A TRANSPORTATION FORMULATION OF FLOW TIME SCHEDULING PROBLEM</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>B PROOF OF THEOREM 1</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>C PROOF OF THEOREM 2</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>D PROOF OF THEOREM 7</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>E PROOF OF EQUATION (4.1)</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>F BRANCH RULE - 2 IN CONJUNCTION WITH PRECEDENCE MATRIX</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>G FLOW CHART OF PROGRAM 4</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>H DESCRIPTION OF VARIABLES IN COMPUTER PROGRAM</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>I FLOW CHART OF BRANCH AND BOUND</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>I.1 Flow Chart of Programs 1-3</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>I.2 Flow Chart of Four Fathoming Rules</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>J TABLES OF 15 PROBLEMS</td>
<td>112</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Interaction Costs in Non-Decreasing Order</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Brute-Force Calculation of Computer Time</td>
<td>54</td>
</tr>
<tr>
<td>6.1</td>
<td>Summarized Computational Results</td>
<td>65</td>
</tr>
<tr>
<td>J.1</td>
<td>List of 15 Problems</td>
<td>112</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Directed Network Flow Diagram</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Description of Branch Rule-1</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Description of Branch Rule-2</td>
<td>52</td>
</tr>
<tr>
<td>5.1</td>
<td>Flow Chart of Dynamic Core Allocation</td>
<td>57</td>
</tr>
<tr>
<td>5.2</td>
<td>Flow Chart of Problem Generation</td>
<td>59</td>
</tr>
<tr>
<td>5.3</td>
<td>Flow Chart of Subroutine ACTL</td>
<td>62</td>
</tr>
<tr>
<td>A.1</td>
<td>Description of Transportation Formulation</td>
<td>78</td>
</tr>
<tr>
<td>A.2</td>
<td>Network Diagram of Transportation Formulation</td>
<td>79</td>
</tr>
<tr>
<td>B.1</td>
<td>Diagram of Schedule S</td>
<td>80</td>
</tr>
<tr>
<td>D.1</td>
<td>Diagram of Schedule S</td>
<td>89</td>
</tr>
<tr>
<td>D.2</td>
<td>Diagram of Schedule S'</td>
<td>90</td>
</tr>
<tr>
<td>D.3</td>
<td>Diagram of Schedule S''</td>
<td>91</td>
</tr>
<tr>
<td>F.1</td>
<td>Diagram of Precedence Relationships Among Jobs</td>
<td>98</td>
</tr>
<tr>
<td>F.2</td>
<td>Diagram of Search Tree</td>
<td>98</td>
</tr>
<tr>
<td>G.1</td>
<td>Flow Chart of Program 4</td>
<td>100</td>
</tr>
<tr>
<td>I.1.1</td>
<td>Flow Chart of Programs 1-3</td>
<td>104</td>
</tr>
<tr>
<td>I.2.1</td>
<td>Flow Chart of Subroutine FATHM 2</td>
<td>108</td>
</tr>
<tr>
<td>I.2.2</td>
<td>Flow Chart of Subroutine FATHM 5</td>
<td>109</td>
</tr>
<tr>
<td>I.2.3</td>
<td>Flow Chart of Subroutine FATHM 3</td>
<td>110</td>
</tr>
<tr>
<td>I.2.4</td>
<td>Flow Chart of Subroutine REMJBP</td>
<td>111</td>
</tr>
</tbody>
</table>
Chapter I

INTRODUCTION

The subject of this dissertation is a methodology for scheduling several one-stage jobs (also termed "tasks" or "activities") on several identical machines (also termed "processors" or "facilities") in parallel in order to minimize the sum of losses associated with the individual jobs. The loss by individual jobs is defined as the cost, which varies linearly with the time a job spends in the system. A scheduling problem with this objective function is often called "The Weighted Flow (or Completion) Time Scheduling Problem."

In general, scheduling problems address the question of allocating limited resources to perform a collection of jobs (tasks). Most scheduling problems are not primary problems but do have secondary importances; scheduling becomes a matter of concern after some fundamental planning decisions have been made. For instance, after market studies and economic analyses for product selection and scale determination are completed and availability of resources is determined, an appropriate schedule for the situation must be planned.
The scheduling problem introduced above arises in a variety of situations, such as, (1) scheduling of classes to classrooms in academic institutions; (2) scheduling of patients on test facilities in hospitals and other health facilities; (3) scheduling of ships to docks in large ports; (4) scheduling of goods to inspectors at inspection stations; (5) scheduling of shipments to a limited number of trucks; and (6) scheduling of aircraft for maintenance to a limited number of hangars.

The general scheduling problem has in essence a combinatorial nature for which an optimal solution can be found by the complete enumeration of all possible solutions. So does a scheduling problem with the optimality criterion of minimizing weighted flow time. Thus an n-job m-machine scheduling problem will have \( (n!)^m \) possible solutions. Clearly, obtaining an optimal solution for such a scheduling problem is often too costly, or even impossible for the large-sized problem, even with the aid of modern computing facilities.

The topic of "The Computational Complexity of Algorithms" is well covered in the literature ([23], [35]). This topic seems to have been started by British mathematician A. M. Turing in the 1930's who had been studying
the mathematical properties of algorithms and was the inventor of the imaginary computer now called a Turing machine.

One result of Turing's work was the conclusion that there are problems in mathematics that can be solved by algorithms and these problems can be classified into two groups: one for which there are efficient, polynomial-time algorithms and the other for which there are only exponential-time algorithms.

An algorithm is called a polynomial-time algorithm if its execution time is bounded by a polynomial function of the problem size. The corresponding problem is said to have a polynomial-time solution. Problems that are known to have polynomial-time solutions are said to be members of the class P (for polynomial). Problems which are not members of the class P are said to be members of the class NP, signifying "Nondeterministic Polynomial". The general n-job m-machine scheduling problem, along with the well-known hard problems, such as knapsack, travelling salesman and clique, have been shown to be members of the NP class ([10],[36]).

The class NP encompasses all the problems in P, or in other words, P is a subset of NP. Only exponential-time algorithms are known for NP class problems as of now. NP class problems may also have polynomial-time
algorithms (in which case NP and P are identical); however, there is strong evidence that NP is not identical to P and NP problems are inherently intractable.

Throughout this dissertation, it is assumed that:

a) Set-up and processing times are independent of the order in which jobs are scheduled, and set-up times, if any, are included in the processing time.

b) All jobs and machines are available at time zero, and processors operate jobs one at a time until all jobs assigned to them are completed.

c) Each job, once started, is not to be interrupted by any other jobs until the job is completed (non-preemption).

d) There is no a priori precedence relationship among jobs, and processing time and weight of each job are deterministically known and remain unchanged throughout the operations.

In the sequel, Chapter 1 presents an introduction of this dissertation including the objective of this research, general descriptions of computational complexity of scheduling problems, practical applications of the Weighted Flow Time Scheduling problem and assumptions made throughout this dissertation.
Chapter 2 is a review of related literature in scheduling problems. First, researches for an overview of general scheduling problems are reviewed and researches for the background of the Weighted Flow Time Scheduling problem are described. Next, general merits and demerits of branch and bound methods, heuristic methods and mathematical programming approaches are discussed, and relevant researches which have been undertaken with those methods are reviewed.

Chapter 3 introduces the problem definition along with five different formulations of the Weighted Flow Time Scheduling problem. Two of those formulations are shown with their possible solution methodologies. Advantages and disadvantages are discussed with respect to five different formulations.

Chapter 4 presents the proposed methodology of the Weighted Flow Time Scheduling problem. The gist of the Branch and Bound Method is discussed as the proposed methodology. Four different fathoming criteria are described as well as two different branching rules. The way of finding an upper bound at the outset is also introduced.

Chapter 5 discusses computer programs developed to implement the proposed methodology and random generation of problems for experimentation. Dynamic core allocations
and brief flow charts of computer programs are also described.

Chapter 6 concludes this dissertation. The results and conclusions are made in this chapter and recommendations for future research are also suggested.

The appendices include most of the proofs needed to conduct this research, detailed flow charts of the computer programs and listings of four different computer programs.
2.1 SURVEY OF SCHEDULING PROBLEMS

In the past two decades there has been a substantial growth in the field of sequencing and scheduling research. An impressive amount of literature, especially on deterministic scheduling problems, has been created.

Elmaghraby [17] exhaustively classified deterministic sequencing problems in terms of the variety, models, context, methodology and current state of the art; in addition, he illustrated the relationship between the theory of sequencing and other areas of control, such as the relationship of sequencing to inventory. Further thorough research for the classification of sequencing literatures has also been done by Day and Hottenstein [13].

Graham, Lawler, Lenstra and Kan [19] completed their research to survey the state of the art of deterministic sequencing and scheduling problems with respect to optimization and approximation algorithms. They
interpreted these algorithms in terms of computational complexity theory. Special cases of single machine scheduling, and identical, uniform and unrelated parallel machine scheduling are also demonstrated in the work of Graham, Lawler, Lenstra and Kan.

Recently, Panwalker and Iskander [33] presented a summary of over 100 different scheduling priority rules, listing many references that analyze them.

Even though a tremendous amount of literature has been developed on deterministic scheduling problems, some problems still remain to be open problems which are currently under heavy attack. One of those problems is the problem of scheduling several one-stage jobs on several machines with the optimality criterion of minimizing weighted flow time. This problem was first introduced by McNaughton [30], but as yet no efficient algorithm has been found for determining an optimal sequencing of jobs due to the complexity of this scheduling problem.

Graham et al. [19] mentioned in their paper that the weighted flow time scheduling problem is NP-hard; hence, it is fruitless to attempt to find polynomial-time optimization algorithms. It is also noted by Baker [1] that the determination of optimal schedules is fairly simple when the weights of all jobs are the same, but surprisingly difficult when weights are not necessarily the same.
Eastman, Even and Isaacs [16] have proposed the best possible lower bound on the Weighted Flow Time Scheduling problem by renumbering jobs according to nondecreasing ratio of $t_i/w_i$, where $t_i$ = processing time of job $i$, and $w_i$ = weight of job $i$. They also introduced a triangular matrix which represents all possible cost elements that contribute to the total cost, and uses the minimum number of cost elements to do so.

Bruno, Coffman and Sethi [7] proposed a non-enumarative solution methodology for the scheduling problem of flow time (where all weights are the same) on $m$ non-identical machines. They showed that the problem can be transformed to the transportation problem. This is detailed in Appendix A.

In this section, brief discussions have been made on the background and the significance of the problem. Next are introduced three methods (branch and bound, heuristic, and mathematical programming) which are very widely used to solve the general scheduling problems.

2.2 **BRANCH AND BOUND**

Branch and bound strategies are among the most widely used methods for combinatorial programming problems. These strategies were first developed and used in the context of mixed integer programming by Land and
Doig [25] and in the traveling salesman problem by Eastman [15]. Soon afterward the applicability of these approaches, also known as "combinatorial programming" or "controlled enumeration," is broadened.

According to Elmaghraby [17], branch and bound strategies are developed on the basis of two principal concepts: (1) the use of a controlled enumeration technique for considering all potential solutions, and (2) the elimination from explicit consideration of particular potential solutions which are known from dominance, bounding and feasibility considerations to be unacceptable. He also restricts the meaning of "combinatorial programming" to problem-solving procedures based on these concepts which are reliable in the sense that when carried to completion, they guarantee the discovery of an acceptable solution if one exists, or the knowledge that none exists.

The idea of a "branch" stems from the fact that in terms of a tree of the problem the procedure attempts to select a branch of the tree to elaborate and evaluate. "Bound" emphasizes the effective use of means for bounding the value of the objective function at each node of the tree, both for eliminating dominated parts and for choosing a branch for elaboration and evaluation.
To obtain an efficient branch and bound algorithm for the weighted flow time scheduling problem, Elmaghraby and Park [18] developed some theoretical results. Their method is compared with our method, and the relative strength of our method is shown in Chapters 4 and 5.

2.3 HEURISTICS

An heuristic algorithm is designed to solve a complicated problem easily and quickly. Although it often yields near-optimal solutions, it is sometimes difficult to determine how far away from optimality the results may be, due to the absence of the theoretical foundation.

Heuristic algorithms emerged to avoid two inevitable disadvantages of the branch and bound, which are typical of implicit enumeration methods. Those disadvantages are as follows. First, the computational requirements are severe for large problems. Second, even for relatively small problems, there is no guarantee that the solution can be obtained quickly since the extent of the partial enumeration depends on the data in the problem. Heuristic algorithms avoid these two drawbacks: they potentially can obtain solutions to large problems with limited computational effort, and their computational requirements are predictable for problems of a given size.
However, heuristic approaches likewise have serious drawbacks: they do not guarantee optimality and, in some instances, their effectiveness is very difficult to judge.

Baker and Merten [2] explored the theoretical properties of the weighted flow time scheduling problem and examined various scheduling procedures in terms of heuristic approaches. In their experiment they considered five different heuristic priority rules such as shortest processing time (SPT), longest processing time (LPT), weighted versions (WSPT and WLPT) and largest weighting factor (W). They reported experimental comparisons of those five rules and concluded that it was difficult to characterize the relative behavior with the identified heuristics.

Coffman and Labetoulle [11] introduced bounds on this problem where all weights are equal to one for the heuristic rules of shortest-rank-first (SRF), longest processing time first (LPT), and RPT (reverse of LPT) with the additional objective of minimum total elapsed time (makespan).

2.4 MATHEMATICAL PROGRAMMING

The mathematical programming approach has been strongly recommended as a possible methodology to solve
scheduling problems. In this dissertation, the mathematical programming approach refers to the set of theories which have collectively come to be known as such, including linear, dynamic, quadratic, and convex programming, integer programming, network of flow, Lagrangian method and the like. In many instances, a numerical answer has not been achieved because of the computational difficulty of the model. This is generally true for all integer programming models, and is also true for some dynamic programming models.

Nevertheless, the proper structuring of the problems in the various models does help in at least two respects. First, the correct formulation of the problems often gives an insight of the problem and also becomes a good basis for further development of solution methodologies. Second, impetus could be given to research in the computing aspects of these mathematical models, which promises to bear fruit in the near future.

The Weighted Flow Time Scheduling problem can be viewed from several different aspects of mathematical formulations.

First, it can be formulated as a program with a quadratic objective function and linear constraints of 0-1 variables. This approach is detailed in Chapter 3.
Second, Belmore, Bennington and Lubore [8] used a network isolation method to solve this problem. The idea behind this method is to isolate nodes from an undirected network. Associated with each edge of the network is a positive cost. This isolation problem is concerned with grouping nodes by cutting edges such that the sum of cut-edge costs is a maximum. Grouping nodes is exactly equivalent to isolating jobs into m groups in our case. And this isolation network problem formulation turns out to be the same form as the quadratic programming problem that is described in detail in Section 4 of Chapter 3.

Dynamic programming techniques have been used to solve a number of scheduling problems. Basically, the technique interprets scheduling and other combinatorial optimization problems as multi-stage decision problems. The main design is to set up recursive equations that describe the optimal criterion function at each stage in terms of information obtained in previous stages.

To solve this problem, Rothkopf [37] presented an algorithm with the dynamic programming approach where each stage corresponds to each job and each state in a stage is represented by a vector, the sum of whose elements equals the sum of processing time of jobs unassigned yet. Since the state vector depends on the
sum of processing time of jobs as of yet unassigned, there will be a tremendous number of states when one deals with jobs having large processing times.

Sahni [38] also presented a dynamic programming type algorithm for this problem when dealing with only two machines. And Baker and Schrage ([3],[41]) presented a dynamic programming approach to solve the special case of the problem when job precedence constraints are known and jobs are assigned to a single machine.

Finally, this problem can also be formulated either as a minimal-cost directed network flow problem or as a set partitioning problem. Either formulation requires extra constraint sets in addition to those of the regular problems. Chapter 3 details their advantages and disadvantages.
Chapter III

PROBLEM FORMULATION

3.1 GENERAL NOTATION

N: \{1, 2, \ldots, n\}, Job indices
M: \{1, 2, \ldots, m\}, Machine indices
\(t_i\): processing time job \(i, i \in N\)
\(w_i\): weight of job \(i, i \in N\)
\(T(i)\): completion time of job \(i, i \in N\)
\(S(i)\): starting time of job \(i, i \in N\)
\(Q_i\): set of jobs to be assigned on machine \(i, i \in M\)
\(T(m_i)\): completion time of machine \(i, i \in M\)
\(C(k)\): cost after \(k\) jobs are assigned
\(LB1(k)\): lower bound-1 after \(k\) jobs are assigned
\(LB2(k)\): lower bound-2 after \(k\) jobs are assigned
\(C_{ij}\): interaction cost when job \(i\) and \(j\) are assigned on
the same machine, which is expressed as \(t_i \cdot w_j\) for
all \(i, j\), where \(i < j\)
3.2 PROBLEM DEFINITION

The weighted flow (completion) time problem can be defined as follows:

\[
\text{min. } \sum_{i \in N} T(i) \cdot w_i \\
\text{s.t. } \bigcup_{i \in M} Q_i = N \\
Q_i \cap Q_j = \emptyset \quad \forall i, j, i \neq j
\]

Therefore the problem can be restated as one of making \( m \) different groups of jobs collectively exhaustive and mutually exclusive in order to minimize the cost of Weighted Flow Time Scheduling problem.

3.3 JOB REORDERING

Jobs are reordered such that

\[
\frac{t_i}{w_i} < \frac{t_j}{w_j} \quad \forall i < j, i, j \in N.
\]

When ties occur, they are broken by placing the job with the smallest processing time first. With this reordering, jobs are in "natural order".

Theorem 1. The Weighted Flow Time Scheduling problem is minimized with the jobs being assigned in natural order when there is only one machine.

Proof: See Appendix B.
Therefore, once jobs are known to be processed on any one of the machines, then the optimal sequence within the machine should be put in natural order by Theorem 1. Throughout this dissertation, jobs are assumed to be in natural order unless otherwise stated.

Lemma 1. There exists an optimal solution in which no jobs wait for processing by any machine while some other machine is idle.

Proof: At an optimal solution, if a job waits for processing by any machine while some other machine is idle, a better solution can be found by assigning that job to the idle machine. Q.E.D.

Next, five different formulations of the Weighted Flow Time Scheduling problem based on mathematical programming approaches are introduced. Solution methodologies are also presented for two different formulations as well as their merits and demerits; and difficulties of developing solution methodologies for the remaining three formulations are also explained.

3.4 FORMULATION 1

The Weighted Flow Time Scheduling problem can be formulated as a 0-1 quadratic programming problem with linear constraints. Formulation is shown in Program P1.

Let \( x_{ij} = 1 \) if job \( j \) is assigned to machine \( i \)
\( x_{ij} = 0 \) otherwise
Program (P1):

\[ \min \sum_{i=1}^{m} \left( \sum_{k=1}^{n} \left( \sum_{j=1}^{k} t_{ij} x_{ij} \right) w_{ik} x_{ik} \right) \]

s.t \( \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in N \)

\( x_{ij} = 0 \) or 1 \( \forall i, j \)

This Program (P1) can be rewritten as Program (P2):

\[ \min Z(\underline{x}) = \sum_{i=1}^{n} \underline{x}_i \cdot Q \cdot \underline{x}_i^T \]

s.t \( \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in N \)

\( x_{ij} = 0 \) or 1

where \( \underline{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \)

\[ Q = \begin{pmatrix} 
\frac{t_1 w_1}{2} & \frac{t_1 w_1}{2} & \cdots & \frac{t_1 w_n}{2} \\
\frac{t_2 w_2}{2} & \frac{t_2 w_2}{2} & \cdots & \frac{t_2 w_n}{2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{t_n w_n}{2} & \frac{t_n w_n}{2} & \cdots & \frac{t_n w_n}{2} 
\end{pmatrix} \]

where \( Q \) is symmetric matrix.

Theorem 2: \( Q \) is positive definite.

Proof: See Appendix C.
Kunzi and Oettle [20] developed an algorithm to solve the above Program P2 when Q is positive definite. Their procedure is to first solve Program P2 with the relaxation of all constraint sets, and to let the solution be $X_o$. They then solve Program P2 only with the relaxation of integer constraints, letting the solution be $X_1$. The idea behind their procedure is to dilate the ellipsoid, relative to $X_o$, until it first passes through an integer point in the convex polyhedron consisting of the constraint set. The iteration procedure proposed by Kunzi and Oettli consists in the dilation not of this ellipsoid, but rather of a polyhedral approximation of it. Steps illustrated by them are as follows. At step $k$, this polyhedron $P_k$ has $k$ faces which are tangent to the ellipsoid, then the polyhedron is dilated until it meets the first lattice point of the given polygonal domain. Next, let $X_{k+1}$ be the lattice point encountered upon dilating $P_k$. Then $X_o$ and $X_{k+1}$ determine a line which meets the ellipsoid $Z(X) = Z(X_1)$ in a point. Call this point $\hat{X}_{k+1}$. Then construct the line (more generally a hyperplane), which is tangent to the ellipsoid at $\hat{X}_{k+1}$, and use this line to form a new approximating polyhedron, $P_{k+1}$, which has $k+1$ faces. The above procedure is repeated until the same $X$ is generated twice in a row, when the problem is solved.
This procedure requires the solution of a mixed integer linear program at every iteration. It seems to be an attractive procedure theoretically, but not practically because the mixed integer linear program itself is a challenging problem to attack.

3.5 LOCAL OPTIMUM OF FORMULATION 1

A very simple algorithm originally proposed by Carlson and Nemhauser [9] acquires a local optimum based on Formulation 1.

In Program P2, due to constraint sets, $\sum_{i=1}^{m} x_{ij} = 1$ for all $j$, $\sum_{i=1}^{n} t_i w_i$ contributes to the objective function as a constant term regardless of the solution set; therefore, one can rewrite Program P2 as Program P3. Program P2 is equivalent to Program P3 except that the value of objective function of Program P2 is as much as $\sum_{i=1}^{n} t_i w_i$ larger than that of Program P3.

Program (P3):

\[
\begin{align*}
\text{min } Z &= \sum_{i=1}^{m} \sum_{j=2}^{n} \sum_{k=1}^{j-1} a_{kj} x_{ij} x_{ik} \\
\text{s.t } \sum_{i=1}^{m} x_{ij} &= 1 \quad \forall j \in N \\
x_{ij} &= 0 \text{ or } 1 \quad \forall i, j \\
\text{where } a_{kj} &= t_k w_j
\end{align*}
\]
Theorem 3. If constraint (3.3) in the above problem is replaced by \( x_{ij} \geq 0 \) \( \forall i, j \), there exists an optimal solution that is an integer solution.

Proof: See [9].

Based on Theorem 3, one can restate the Program P3 without the integer constraints, such that

Program (P3):

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{m} \sum_{j=2}^{n} \sum_{k=1}^{j-1} a_{kj} x_{ij} x_{ik} & (3.4) \\
\text{s.t} & \quad \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in N & (3.5) \\
& \quad x_{ij} \geq 0 \quad \forall i, j & (3.6)
\end{align*}
\]

Theorem 4. Necessary conditions for \( x_{ij} \) to minimize (3.4) subject to (3.5) and (3.6) are that there exist scalars \( \lambda_j \) such that

\[
\begin{align*}
\text{where} & \quad r_{ij} = \lambda_j \quad \text{when} \quad x_{ij} > 0 & (3.7) \\
& \quad r_{ij} = \sum_{k} a_{kj} x_{ik} \quad \forall i, j & (3.8)
\end{align*}
\]

\[
\begin{align*}
\text{where} & \quad r_{ij} = \lambda_j \quad \text{when} \quad x_{ij} > 0 & (3.9) \\
& \quad x_{ij} \geq 0 \quad \forall i, j \\
& \quad \sum_{i} x_{ij} = 1 \quad \forall j
\end{align*}
\]
Proof: The above are Kuhn-Tucker conditions with $\lambda_j$ as Lagrange Multipliers.

Theorem 5. A sufficient condition for a solution that satisfies (3.5), (3.6), (3.7), and (3.9) to be a local minimum is that at most n of the inequalities $r_{ij} - \lambda_j \geq 0$ are satisfied as equalities.

Proof: See [9].

Theorem 6. A change of $2(r_{qs} - \lambda_s)$ in the objective function results from a shift of assignment of job s from machine p to machine q.

Proof: See [9].

Clearly from Theorem 6, to make maximum improvement in the objective function with a single shift, one must find only the

$$\max_{ij}(\lambda_j - r_{ij})$$

Based on Theorems 3, 4, 5 and 6, the following steps generate a local optimum solution of Formulation 1.

Step 1) If the maximum of $(\lambda_j - r_{ij})$ is positive and occurs for $i = q$ and $j = s$ with $\lambda_s = r_{ps}$, then the shift is from $x_{ps} = 1$ to $x_{qs} = 1$. 

Step 2) If the maximum of \((\lambda_j - r_{ij})\) is zero and
\[ \lambda_j = \min_{i} r_{ij} \] is unique in each row, no further improvement can be made with a single shift.
The present solution is a local minimum.

Step 3) If the maximum is zero and \( \lambda_j = \min_{i} r_{ij} \) is not unique in at least one row, in other words,
\[ \lambda_s = r_{ps} \text{ and } r_{ps} = r_{qs}, \] make the shift from
\[ x_{ps} = 1 \text{ to } x_{qs} = 1 \] which may lead to Step (1) at the next step. If an unsuccessful attempt has been made to break all "ties," one can only conclude that all the tie solutions correspond to local minima.

This procedure can easily be illustrated with a small example of 5 jobs and 2 machines.

Example: 5 jobs and 2 machines

<table>
<thead>
<tr>
<th>job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_i )</td>
<td>5</td>
<td>21</td>
<td>16</td>
<td>6</td>
<td>26</td>
</tr>
<tr>
<td>( w_i )</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>( t_i/w_i )</td>
<td>1.25</td>
<td>4.2</td>
<td>5.3</td>
<td>6</td>
<td>6.5</td>
</tr>
</tbody>
</table>
Then \( Q = \begin{bmatrix}
0 & 12.5 & 7.5 & 2.5 & 10 \\
12.5 & 0 & 31.5 & 10.5 & 42 \\
7.5 & 31.5 & 0 & 8 & 32 \\
2.5 & 10.5 & 8 & 0 & 12 \\
10 & 42 & 32 & 12 & 0
\end{bmatrix} \)

1. A possible feasible solution with its corresponding \( R \) matrix is

\[
X^T = \begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1 \\
0 & 1 \\
0 & 1
\end{bmatrix}, \quad R^T = \begin{bmatrix}
12.5^* & 20 \\
12.5^* & 84 \\
39.0 & 40^* \\
13.0 & 20^* \\
52 & 44^*
\end{bmatrix}
\]

where machine 1: Job1, Job2
machine 2: Job3, Job4, Job5

In the \( R^T \) matrix, the values of \( r_{ij} \) for which the corresponding \( x_{ij} = 1 \) are starred (\( * \)). Then \( \max_{ij} (\lambda_j - r_{ij}) = 20 - 13 = 7 = (\lambda_4 - r_{41}) \). Then, shifting job 4 to machine 1 will reduce that cost of \( 7 \times 2 = 14 \),
Since \( \lambda_j = \min_i r_{ij} \) for all \( i \) and is unique in each row of \( R^T \), a local minimum has been achieved, where

- machine 1: Job1, Job2, Job4
- machine 2: Job3, Job5

As is described above, one can only use this algorithm when the objective function matrix \( Q \) has diagonal elements equal to zero. When all diagonal elements are set to zero, the matrix \( Q \), already proved to be positive definite, becomes indefinite because all off-diagonal elements are strictly positive. Since \( Q \) turns out to be indefinite, implementing this algorithm generates only local optimum.

Greenberg [22] contributes an algorithm to solve Program P3, however, it does not seem to be a good procedure since it is actually a total enumeration.
3.6 **FORMULATION 2**

The Weighted Flow Time Scheduling problem can also be formulated as a minimal-cost directed network flow problem. This formulation is explained more succinctly with a simple example involving 3 jobs and m machines, where \( m < 3 \) to make the problem interesting. The directed network flow is shown in Figure 3.1,

![Directed Network Flow Diagram](image)

**Figure 3.1. Directed Network Flow Diagram.**
where the source has \( m \) units to be transferred to the sink and each arc represents the corresponding cost and has a maximum capacity of 1 unit,

and

\[ \text{notation } (i, j) \Rightarrow \]

\( i: \) sequential node number
\( j: \) job number for the node

Then the problem is to find \( m \) different paths from source to sink in order to minimize the total cost such that all jobs are processed once and only once.

Here, the number of nodes required with this formulation is \( \frac{1}{6}n(n - 1)(n + 1) + n \) and the number of arcs required is \( \frac{1}{3}n(n - 1)(n + 1) + 2n \).

This formulation is very hard to implement, because costs are all path-dependent and an additional constraint is required to ensure that all jobs are processed only once.

3.7 FORMULATION 3

The Weighted Flow Time Scheduling problem can also be formulated as a set partitioning problem. Formulation is given in Program (P4).
Program (P4):

\[
\begin{align*}
\min Z &= \sum_{j=1}^{q} g_j x_j \quad \cdots \cdots \cdots (3.10).
\end{align*}
\]

\[
\begin{align*}
s &\cdot t \sum_{j=1}^{q} a_{ij} x_j = 1, \quad i = 1, \ldots, n \quad (3.11)
\end{align*}
\]

\[
\begin{align*}
\sum_{j} x_j &= m, \quad j = 1, \ldots, q \quad (3.12)
\end{align*}
\]

\[
\begin{align*}
x_j &= 0 \text{ or } 1
\end{align*}
\]

where \( q = \sum_{i=1}^{n} C_i^n = (2^n - 1) \)

\( a_{ij} \) is either 1 or 0

and \( g_j \) is the cost when all jobs in set \( S_j \)

are assigned on the same machine where

\[
S_j' = \{ i \mid a_{ij} = 1 \}.
\]

Compared to a regular set partitioning problem, however, this formulation has one additional constraint set (3.12). This approach is not attractive to implement because of the exponential increase in the number of columns, \( q \) with respect to the number of jobs, and the extra constraint set (3.12).
3.8 FORMULATION 4

The Weighted Flow Time Scheduling problem can also be formulated as a dynamic programming problem. The method presented here was originally proposed by Rothkopf [37].

If the jobs are ordered in reverse of natural order, the following condition must be satisfied in the optimal solution [37]:

\[ S(i) \leq \sum_{j=i+1}^{n} t_j \quad \forall i \in N \]

Then each stage can be referred to each job, and each state of a stage \( i \) can be referred to each element of set \( T_i \), where

\[ T_i = \left\{ (l_1, l_2, \ldots, l_m) \right\} \left\| \sum_{k=1}^{m} l_k = \sum_{j=i+1}^{n} t_j, l_k \text{ is nonnegative integer for } k \in M \right\} \]

where \( T_i \) is not an ordered set. Then the recursive equation is

\[ R_i(L_i) = \min_{1 \leq j \leq m} \left[ R_{i-1} \left( L_i + e_i t_i \right) + w_i (t_i + l_j) \right] \]

where \( L_i = (l_1, l_2, \ldots, l_m) \in T_i \)

and \( e_j \) is the vector with the \( j^{th} \) element equal to one and all other components equal to zero and set \( R_0 = \emptyset \).
Once one solves the problem with the above recursive equation, one can trace back to the first stage in order to determine the optimal allocation of jobs to the machines, according to $L_i^*$, where $L_i^*$ is equal to $L_i$ whose $R_i(L_i)$ gives the optimal in the recursive equations. Since the state vector $T_i$ from Equation (3.13) depends on the sum of processing times of jobs as of yet unassigned, this procedure would require a tremendous number of states if one deals with jobs having large processing times.

3.9 FORMULATION 5

Finally, the Weighted Flow Time Scheduling problem can be formulated as 0-1 linear programming problem. Formulation is given in Program P5.

Let $y_{ij} = 1$ if jobs $i$ and $j$ are assigned on the same machine

$= 0$ otherwise

$C_{ij} = t_i w_j$: interaction cost, where $i < j$
Program (P5):

\[
\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij}y_{ij}
\]

s.t. \[
\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} y_{ij} \geq K \quad \ldots \ldots \quad (3.14)
\]

\[
y_{ij} + y_{ik} - y_{jk} \leq 1 \quad i = 1, \ldots, n-2 \quad (3.15)
\]

\[
y_{ij} - y_{ik} + y_{jk} \leq 1 \quad j = j+1, \ldots, n-1
\]

\[
-y_{ij} + y_{ik} + y_{jk} \leq 1 \quad k = j+1, \ldots, n
\]

\[
|\tilde{N}| + |S| = m \quad \ldots \ldots \quad (3.16)
\]

\[
y_{ij} = 0 \text{ or } 1 \quad i, j
\]

where \( K = P_1 \left( \frac{1}{2}(L + 1) \right) L + \frac{1}{2} P_2 (L)(L - 1) \)

and \( L = \lceil n/m \rceil \); the largest integer less than or equal to \( n/m \) and \( P_1 = n - L \cdot m \), \( P_2 = m - P_1 \)

\( S = \{(s_1, \ldots, s_q) | s_i \text{ is set of jobs assigned on the same machine and } \min_{i=1\ldots q} |s_i| \geq 2\} \)

\( \hat{N} = N - \bigcup_{i=1}^{q} S_i \)
Here, the constraint (3.16) makes this formulation hard to implement. However, this formulation is used as one of the fathoming criteria in the Branch and Bound Method shown in Chapter 4.

In this chapter, five different formulations of the Weighted Flow Time Scheduling problem have been presented. Two of them have been presented with their solution methodologies. And three of them have been discussed with the explanations of why it is very hard to implement those formulations.
Branch and Bound Method

The gist of the Branch and Bound Method is to try to limit the explicitly-enumerated nodes in the search tree by feasibility, dominance and lower bound argument, and also by establishing precedence relations among jobs so that it is not necessary to search for an optimal schedule over all possible schedules, but only over a subset of schedules. This chapter describes several features of the branch and bound procedure including fathoming rules, determination of lower bounds and branching schemes.

4.1 FATHOMING RULES

4.1.1 Precedence Matrix

There is no a priori precedence relationship among jobs, as described in Assumption (d) in Chapter 1. However, implied precedences among jobs could be deduced from considerations of the features of an optimal solution.
Define "job i precedes job j" in a schedule means that the starting time of job i is less than or equal to the starting time of job j.

Theorem 7. If \( t_i < t_j \) and \( w_i > w_j \), then job i will not start later than job j in an optimal solution.

Proof: See Appendix D.

Corollary 1: Starting time of job 1 precedes all other jobs in an optimal solution.

Proof: If any job \( i(\neq 1) \) precedes job 1 in an optimal solution, the starting time of job 1, \( S(1) \), is strictly positive. Assume job 1 is on machine k. Then, two cases are considered:

Case (1) Since \( S(1) > 0 \), there exists a job before job 1 for processing on machine k, which is contrary to an optimal solution by Theorem 1.

Case (2) Since \( S(1) > 0 \), machine k will experience idle time before job 1 is processed, which contradicts the conditions for an optimal solution by Lemma 1. Q.E.D.

Then a precedence matrix (PREM) can be constructed based on Theorem 7.
Let \( \text{PREM}(i,j) = 1 \) if job \( i \) precedes job \( j \) where \( i < j \)
\( \text{PREM}(i,j) = 0 \) if no precedence of job \( i \) relative to job \( j \)

Then, immediate successor(s) and immediate predecessor(s) of each job can be determined by the following procedure.

**Steps for the determination of the immediate successor(s) and predecessor(s)**

- **Step 1**, set \( i = 0 \)
- **Step 2**, \( i = i + 1 \), if \((i > (n - 2))\) go to **Step 7**
- **Step 3**, \( j = i \)
- **Step 4**, \( j = j + 1 \), if \((j > (n - 1))\) go to **Step 2**
  - if \( \text{PREM}(i,j) = 0 \), go to **Step 4**
  - otherwise, go to **Step 5**
- **Step 5**, \( k = j \)
- **Step 6**, \( k = k + 1 \), if \((k > n)\) go to **Step 4**
  - if \( \text{PREM}(i,k) = 1 \) and \( \text{PREM}(j,k) = 1 \), then set \( \text{PREM}(i,k) = 0 \)
  - otherwise go to **Step 6**
- **Step 7**, stop.
As a result of the above procedure,

\[ \text{PREM}(i,j) = 1 \quad \text{if job } i \text{ is immediate predecessor of} \]
\[ \text{job } j \text{ (or job } j \text{ is immediate successor of job } i) \]
\[ = 0 \quad \text{otherwise} \]

Let define "\( V_i < V_j \)" to mean that all jobs in set \( V_i \)
strictly precede all jobs in set \( V_j \).

By Theorem 7, one can identify successive sets of jobs which strictly precede the remaining sets of jobs, such as

\[ V_1 < V_2 < \ldots \ldots < V_P \]

where \( V_i \cap V_j = \emptyset \quad \forall \, i \cdot j \in P = \{1, 2, \ldots, P\}, \, i \neq j \)

\[ \bigcup_{i \in P} V_i = N \]

Corollary 2: If \( |V_1| \leq m \), jobs in \( V_1 \) should not be assigned on the same machine in the optimal solution.

Proof: Suppose two jobs, \( i \) and \( j \), in \( V_1 \) are assigned to the same machine and \( i < j \). Then, \( S(i) < S(j) \)
and \( S(j) < S(k) \) for \( k \in V_2 \cup \ldots \cup V_P \). Since
\[ |V_1| \leq m, \text{ there exists at least one machine that experiences an idle time of } S(j) - s(i) > 0. \] By Lemma 1, this cannot be the optimal solution.

Hence, the proof is completed.

Corollary 3: While assigning a job \( i \in V_{p'}, p' \in P \), if all jobs in \( V' = \bigcup_{i=1}^{p'} V_i \) except job \( i \) are assigned already, then job \( i \) should be assigned to the earliest available machine in the optimal solution.

Proof: When all jobs in \( V' = \bigcup_{i=1}^{p'} V_i \), except job \( i \), are assigned, then let \( T(m_{k^*}) = \min_{k \in M} T(m_k) \). If job \( i \) is assigned to any machine \( j \neq k^* \), then \( S(i) > T(m_{k^*}) \) since \( S(i) \leq S(1) \) for \( l \in V_{p'+1} \cup \ldots \cup V_p \) machine \( k^* \) will experience an idle time \( S(i) - T(m_{k^*}) > 0 \). Again this solution is not optimal by Lemma 1. The proof is completed.

Here Corollary 2 and Corollary 3 will be used in Branching Rule-1, which will be described in Section 4.2 of this chapter. As the Branch and Bound Method proceeds, whenever the precedece relationships among jobs described in this section are violated, this branch does not lead to an optimal solution. Therefore fathoming should take place.


4.1.2 Lower Bound-1

Let $C_{m,n}$ be the cost of a Weighted Flow Time Scheduling problem with $m$ machines and $n$ jobs. Then Eastman, Even and Isaacs [16] developed the best possible lower bound expressed in terms of $m$, $C_{n,n}$ and $C_{1,n}$. Their lower bound is as follows.

$$C_{m,n} \geq \left(\frac{1}{m}\right) \cdot \left(\frac{1}{C_{1,n}} - \frac{1}{2}C_{n,n}\right) + \frac{1}{2}C_{n,n} \quad (4.1)$$

Proof is given in Appendix E.

A modification of the bound in Equation (4.1) is used in the Branch and Bound method presented here. Let $T_{\text{min}}$ be the earliest starting time among available machines after $k$ jobs are assigned, and let $C_{i,j}(T)$ be the cost when there are $i$ machines and $j$ jobs with all machines being available at time $T$. Then the lower bound for the remaining $(n - k)$ jobs, after $k$ jobs are assigned to the machines, is as follows.

Corollary 4:

$$L_B(k) = C(k) + \frac{1}{m} C_{1,n-k}(T_{\text{min}}) + \frac{m-1}{2m} C_{n-k,n-k}(T_{\text{min}})$$

$$+ T_{\text{min}} \sum_{j \in (N-k)} w_j$$

Proof: From Equation (4.1),
Here one conservatively assumes that $T_{\text{min}}$ is the earliest available time of all the machines for the remaining $(n - k)$ jobs.

Since $T \geq T_{\text{min}}$, therefore,

$$C_{m,n-k}(T) \geq C_{m,n-k}(T_{\text{min}}) \geq \frac{1}{m}(C_{1,n-k}(T_{\text{min}}) - \frac{1}{2}C_{n-k,n-k}(T_{\text{min}}))$$

$$+ \frac{1}{2}C_{n-k,n-k}(T_{\text{min}}) + T_{\text{min}} \sum_{j \in (N-k)} w_j$$

Here $C(k)$ is a given constant cost for the current assignment of $k$ jobs.

Therefore,

$$LB_1(k) = C(k) + C_{m,n-k}(T_{\text{min}})$$

$$= C(k) + \frac{1}{m}(C_{1,n-k}(T_{\text{min}}) + \frac{m-1}{2m}C_{n-k,n-k}(T_{\text{min}}))$$

$$+ T_{\text{min}} \sum_{j \in (N-k)} w_j$$

This completes the proof.
A lower bound $L_{Bl}(k)$ will be used as one of the fathoming rules at any partial scheduling of $k$ jobs in the Branch and Bound method presented here. Whenever the value of $L_{Bl}(k)$ is greater than or equal to a current incumbent solution, this branch does not lead to an optimal solution. Therefore fathoming should take place.

4.1.3 Lower Bound-2

This bound is basically derived from Formulation 5 in Chapter 3. There, $C_{ij}'$ represented the interaction cost which occurred when job $i$ and $j$ are assigned to the same machine. Here, $C_{ij}'$ are listed in nondecreasing order and reindexed with a single subscript for notational convenience such that $C_i \leq C_j$ for all $i < j$ (see Table 4.1).

Let $K^*$ be the minimum number of interaction terms, then

$$K^* = \frac{1}{2} P_1 \cdot (L + L) \cdot (L) + \frac{1}{2} P_2 \cdot (L) \cdot (L - 1)$$

where $L = \lfloor n/m \rfloor$, $\lfloor \rfloor$ is the largest integer less than or equal to the value of $n/m$ and

$P_1 = n - L \cdot m$, $P_2 = m - P_1$

Then the lower bound is as follows:
Corollary 4:

\[ LB2(0) = \sum_{i=1}^{K^*} C_i + \sum_{i=1}^{n} t_i w_i \]

Proof: \( LB2(0) \) is the solution of Program P5 with the constraints (3.15) and (3.16) relaxed. Q.E.D.

Now, after \( k \) jobs are assigned, \( K^* \) should be updated whenever any one of the following conditions holds:

- **Condition 1**: \( |Q| > L + 2 \) if \( P \neq 0 \)
- **Condition 2**: \( |Q| > L + 1 \) if \( P = 0 \)
- **Condition 3**: number of \( |Q| \) whose value is greater than \( L + 1 \) is greater than \( P \) when \( P \neq 0 \)

where \( |Q| \) is the cardinal number of set \( Q \).

When any of the above Conditions (1-3) holds, \( K^* \) should be updated as follows:

<K* updating steps>

1. **Step 1**: Set \( SUM = 0, N1 = n, M1 = m, i = 0 \) and \( ind = 0 \)
2. **Step 2**: \( i = i + 1, \) if \( (i > m) \) STOP
   - if \( P = 0 \) and \( |Q| \leq L \), go to Step 2
   - if \( P = 0 \) and \( |Q| > L + 1 \), go to Step 3
   - if \( P \neq 0 \) and \( |Q| \leq L \), go to Step 2
   - if \( P \neq 0 \) and \( |Q| > L + 1 \), then \( ind = ind + 1 \)
if \( \text{ind} \leq P_1 \) and \( |Q_1| = L + 1 \), go to Step 2
otherwise, go to Step 3

Step 3

\[
\text{SUM} = \text{SUM} + \frac{1}{2}(|Q_1|)(|Q_1| - 1)
\]

\[
M_1 = M_1 - 1
\]

\[
N_1 = N_1 - |Q_1|
\]

go to Step 2

Then,

\[
K^* = \frac{1}{2}P_1 \cdot (L + 1) \cdot (L) + \frac{1}{2}P_2 \cdot (L) \cdot (L - 1) + \text{SUM}
\]

where \( L = \lceil N_1/M_1 \rceil \)

\[
P_1 = N_1 - L \cdot M_1 \text{ and } P_2 = M_1 - P_1
\]

Also, let \( Z_D \) be the number of \( C_i \), for \( i \leq K^* \), whose interaction cost never contributes to the total cost, and \( O_U \) be the number of \( C_i \), for \( i > K^* \), whose interaction cost contributes to the total cost for any partial schedule of \( k \) jobs.

Then,

\[
\text{LB}_2(k) = C(k) + \sum_{i=1}^{K^*} C_i + (\ell_1) \cdot \text{SUM}(\ell_3^+) - (\ell_2) \cdot \text{SUM}(\ell_3^-) + \sum_{i=1}^{n} w_i t_i \quad \ldots \quad (4.2)
\]

where \( \ell_3 = |O_U - Z_D| \), and
SUM($\ell_3^+$) : Sum of $C_{K^*+1}$ and the next $\ell_3-1$ values of $C_i$ between $i = K^* + 1$ and $i = n$ corresponding to jobs as yet unassigned.

SUM($\ell_3^-$) : Sum of $C_{K^*}$ and the next $\ell_3-1$ values of $C_i$ between $i = K^*$ and $i = 1$ corresponding to jobs as yet unassigned.

if $OU - ZD < 0$, $\ell_1 = 1$ and $\ell_2 = 0$
if $OU - ZD > 0$, $\ell_1 = 0$ and $\ell_2 = 1$
if $OU - ZD = 0$, $\ell_1 = \ell_2 = 0$

$C_i$ is the interaction cost yet undecided with this partial schedule whether to contribute to the total cost.

Example: Consider a problem of 5 jobs and 2 machines.
Table 4.1 is constructed based on nondecreasing order of interaction costs.
Table 4.1 Interaction Costs in Non-decreasing Order

<table>
<thead>
<tr>
<th>Interaction cost</th>
<th>Jobs*</th>
<th>Reindexed**</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{12}$</td>
<td>1 and 2</td>
<td>$C_1$</td>
</tr>
<tr>
<td>$C_{15}$</td>
<td>1 and 5</td>
<td>$C_2$</td>
</tr>
<tr>
<td>$C_{13}$</td>
<td>1 and 3</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$C_{24}$</td>
<td>2 and 4</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$C_{25}$</td>
<td>2 and 5</td>
<td>$C_5$</td>
</tr>
<tr>
<td>$C_{34}$</td>
<td>3 and 4</td>
<td>$C_6$</td>
</tr>
<tr>
<td>$C_{14}$</td>
<td>1 and 4</td>
<td>$C_7$</td>
</tr>
<tr>
<td>$C_{35}$</td>
<td>3 and 5</td>
<td>$C_8$</td>
</tr>
<tr>
<td>$C_{23}$</td>
<td>2 and 3</td>
<td>$C_9$</td>
</tr>
<tr>
<td>$C_{45}$</td>
<td>4 and 5</td>
<td>$C_{10}$</td>
</tr>
</tbody>
</table>

* If jobs assigned on the same machine.

** $C'_{ij}$ is reindexed with single subscript such that $C_1 \leq C_j$ for $i < j$, i.e., $C_1 = C'_{12}$, $C_2 = C'_{15}$ and $C_1 \leq C_2$, etc.
Then consider the following partial schedule:
machine 1: Job1 and Job3
machine 2: Job2 and Job4

For this partial schedule, $K^* = 4$

Since Jobs 1 and 3 are on the same machine, and
Jobs 2 and 4 are on the same machine, $C_{13} = C_3$ and
$C_{24} = C_4$ contributes to the total cost; however Jobs 1
and 2 are on different machines, thus $C_{12} = C_1$ never
contributes to the total cost. Neither do $C_{14} = C_7$,
$C_{23} = C_9$ and $C_{34} = C_6$.

Therefore, from Equation (4.2),

\[ C(k) = C_3 + C_4 \]

\[ \sum_{i=1}^{K^*} C_i = C_2 \]

OU = 0

ZD = 1

\[ \ell_3 = |0 - 1| = 1 \]

Since ZD > OU, \( \ell_1 = 1 \) and \( \ell_2 = 0 \)

\[ (\ell_1) \sum_{i=K^*+1}^{\ell_3} C_i = C_5 \]

Therefore,

\[ LB2(k = 4) = (C_3 + C_4) + C_2 + C_5 + \sum_{i=1}^{5} w_i t_i \]
A lower bound $LB2(k)$ will be used as one of the fathoming rules at any partial scheduling of $k$ jobs in the Branch and Bound method presented here. Whenever the value of $LB2(k)$ is greater than or equal to a current incumbent solution, this branch does not lead to an optimal solution. Therefore fathoming should take place.

4.1.4 Earliest Starting Time

After job number $k_1$ is assigned, let $I$ be the set of unassigned jobs of no precedence relations with job $k_1$, and let $E_i$ be the earliest possible starting time of job $i \in I$. $E_i$ can be derived from PREM$(i,j)$ for $i \in I$. Also, let $\{NM\}$ be the number of machines that have the earliest starting time of available machines, $\{NJ\}$ be the number of jobs among set $I$ of which the earliest possible starting time is less than or equal to the earliest starting time of available machines.

Corollary 5: For a partial schedule, neither one of the following two cases ever lead to an optimal solution.

Case 1) $\{NM\} = 1$, and $E^*_i = \min_{i \in I} E_i$ is greater than the earliest starting time of available machines.

Case 2) $\{NM\} \geq 2$ and $\{NJ\} < \{NM\}$
Proof: Case 1) Let $T(m^*_i) = \min T(m_i)$. Since $E^*_i > T(m^*_i)$, machine number $i^*$ will experience an idle time at least of value $E^*_i - T(m^*_i) > 0$ which would not lead to the optimal solution by Lemma 1.

Case 2) Let $T(m^*_k) = \min T(m_k)$ since $\{NJ\} < \{NM\}$, at least $\{NM\} - \{NJ\}$ number of machines will experience an idle time of value strictly greater than zero, which again by Lemma 1 would not lead to the optimal solution.

This completes the proof.

As the Branch and Bound proceeds, conditions of Case (1) and Case (2) in Corollary 5 are checked. When either Case (1) or Case (2) is realized, this branch would not lead to an optimal solution. Therefore fathoming should take place.

4.2 BRANCHING RULES

One can briefly state two basic concepts of implicit enumeration by Branch and Bound methods. One is that all feasible schedules must be accounted for either explicitly or implicitly, and the other is that the least number of schedules should be accounted for explicitly. Therefore,
one tries to eliminate as many partial schedules as possible, while accounting for all partial schedules implicitly.

In the search tree generation, backtracking rather than jumptracking is used. Hence, improved upper bounds are more easily generated as the search proceeds, and computer storage required for backtracking is much smaller than that of jumptracking.

The efficiency of a branch and bound algorithm depends upon both the branching scheme used to build a search tree and also the fathoming rules incorporated in the algorithm.

This section presents two branching rules. Branch rule-1 is based on the fact that, because of natural ordering, the total possible number of schedules of all jobs on m machines is $m^n$. Branch rule-2 is based on the fact that the total possible number of schedules of all jobs is $n!$, regardless of the number of machines.

Branch rule-1, which is new in the context of the Weighted Flow Time Scheduling problem, is more attractive than Branch rule-2, which has been implemented by Elmaghraby and Park [18], because $m^n$ is much less than $n!$ since $m$ is much less than $n$ for most practical problems. Next, Branch rule-1 and Branch rule-2 are described with small examples for a clearer understanding.
4.2.1 **Branch rule-1**

With Branch rule-1, each node in the tree generally has m different descendants at the next level. Each of these descendants corresponds to a machine. Each level corresponds to a job. The Branch rule-1 can easily be explained with the following small example.

Example: Consider a 5 job, 2 machine problem. The branching scheme is as shown in Figure 4.1.

* A tree is defined as a set of arcs if it satisfies two conditions: 1. The arcs generate a connected subgraph; 2. The arcs contain no cycles.
In Figure 4.1, at node 2, which is reached if job 1 is assigned to machine 1, there are two descendants represented by nodes number 3 and 8. If job 2 is assigned to machine 1, it will lead to node number 3, and if job 2 is assigned to machine 2, it will lead to node number 8. Similarly, there are two descendants from node number 9 corresponding to job 2's allocation to machine 1 or 2, given that job 1 was assigned to machine 2 and so on.

The total number of nodes to be explored with this branching scheme, assuming no fathoming, is \( n \sum_{k=1}^{m} m^k \). The order in which jobs are assigned within a machine is the natural order.

4.2.2 Branch rule-2

This branching rule can be stated as follows: assign one job at a time on the earliest available machine until all jobs are assigned. The order in which jobs are assigned is any one of the possible \( n! \) sequences.

The Branch rule-2 can easily be explained with the following example.

Example: Consider a 5 job and 2 machine problem. The resulting branching scheme is shown in Figure 4.2. In Figure 4.2, at node 1, five descendants are represented by nodes number 2, 3, 4, 5, and 6. If job 1 is assigned on the earliest
available machine, it will lead to node number 2. And then if job 2 is assigned on the earliest available machine, it will lead to node number 7. Similarly, if job 3 is assigned on the earliest available machine after job 1 is assigned, it will lead to node number 8, and so on.

Figure 4.2. Branch Rule-2

The total number of nodes to be generated with this branch rule, assuming no fathoming, is \( \sum_{r=1}^{n} \frac{n!}{(n-r)!} \).
The implementation of Branch rule-2 becomes more efficient in conjunction with the precedence matrix developed in Section 4.1 of Chapter 4. This is detailed in Appendix F.

Two different branching schemes have been presented in this section. Based on Branch rule-1, the total number of schedules of all jobs on $m$ machines is $m^n$, and based on Branch rule-2, the total number of schedules of all jobs is $n!$ regardless of the number of machines. Branch rule-1, which is new, seems to have more strength than Branch rule-2, because $m^n$ is less than $n!$ for the most practical problems.

4.3 UPPERBOUND AND BRUTE-FORCE CALCULATION OF COMPUTER TIME

At the outset, a solution obtained by assigning jobs in natural order one at a time to the earliest available machine is used as an upper bound. This upper bound will be updated whenever a better complete solution is generated as the search proceeds.

Here brute-force calculations of computer time required to generate a number of nodes assuming that no fathoming takes place are presented in Table 4.2. It is also assumed, based on Table 6.1, that generating 500 nodes, which includes all updating times of node information, will take .10 second.
Table 4.2 Brute-Force Calculation of Computer Time

<table>
<thead>
<tr>
<th># jobs</th>
<th>n = 15</th>
<th>n = 20</th>
<th>n = 25</th>
</tr>
</thead>
<tbody>
<tr>
<td># node*</td>
<td>total # nodes</td>
<td>C.P.U.</td>
<td>total # nodes</td>
</tr>
<tr>
<td>n!</td>
<td>1.3 x 10^{12}</td>
<td>8.3 yrs.</td>
<td>2.4 x 10^{18}</td>
</tr>
<tr>
<td>2^n</td>
<td>3.2 x 10^{4}</td>
<td>6.6 sec.</td>
<td>1.1 x 10^{6}</td>
</tr>
<tr>
<td>3^n</td>
<td>1.4 x 10^{7}</td>
<td>0.8 hrs.</td>
<td>3.5 x 10^{9}</td>
</tr>
</tbody>
</table>

* Assuming that n!, 2^n, and 3^n number of nodes are required to exhaust the enumeration without any fathoming.
5.1 DYNAMIC CORE ALLOCATION

The computer program developed in this dissertation was coded in FORTRAN Language and was run on an Amdahl 470 using IBM software. This program has a feature of dynamic core allocation which is realized by having access to two subroutines GETM and FREEM in FORTLIB, one of IBM's software packages. Call to subroutine GETM allocates the amount of core specified for each indicated array. It also re-executes the program prologue, thereby updating the compiler-generated addresses for the argument arrays as well as for some compiler-generated constants. A call to subroutine FREEM releases the storage allocated to the specified arrays. Some of the restrictions in using subroutine GETM and FREEM are as follows:

a) GETM may be called only from the FORTRAN subroutine subprogram.

b) Only the names of the dynamically allocated arrays may appear in the subroutine declaration of these calling programs, and they have to
appear in the same order of the subroutine declaration as in the call to GETM.

c) Dimension variables must be placed in a common block.

A flow chart of the dynamic core allocation procedure is shown in Figure 5.1.

5.2 PROBLEM GENERATION

Problems are generated by calling subroutine KERAND in FORTLIB, one of the IBM software packages. A call to KERAND is made to compute a uniformly distributed random real number on (0,1). Once a random number in the interval (0,1) is generated by calling subroutine KERAND, then it is multiplied by 100 and truncated after the decimal point to produce a two-digit random number. This integer value is used as the processing time of a job. The above procedure is repeated to generate another value of the weight of a job, and so on. A flow chart for the problem generation is shown in Figure 5.2.

For the experiment, 15 different problems ranging from a 5-job problem to a 30-job problem are generated and are shown in Appendix J.
Flow Chart of Dynamic Core Allocation

main program

CALL SUB

STOP

Subroutine SUB

Input data
N = # jobs
ITLNM = # machines

Calculation of necessary arrays to be allocated in the storage

Call GETM: Allocation of the amount of core specified for each previously calculated array

Call ATCL: Actual program for Branch and Bound method detailed in Appendix
Figure 5.1 Flow Chart of Dynamic Core Allocation

Call FREEM: Release of the arrays allocated to the specified storage

RETURN
Flow Chart of Problem Generation

Initialization
ISEED = 999
K1 = ISEED

i = 1

CALL KERAND (K1, K2, F)

Set IZ to the processing time of job i
IZ = F * (10 ** 2)

CALL KERAND (K1, K2, F)

Set IK to the weight of job i
IK = F * (10 ** 2)

Figure 5.2 Flow Chart of Problem Generation
To generate those 15 problems, one must maintain the same seed number. Keeping the same seed number for the generation of the whole problem set is an attempt to avoid any influence of different problem characteristics (due to a different seed number) on the measure of relative performances of fathoming rules or branching rules.

5.3 PROGRAM DEVELOPMENT

The following four different fathoming rules discussed in Chapter 4 are used in the program to decide whether to fathom.

Fathoming rule - 1 is based on the precedence matrix and is implemented by calling subroutine FATHM2.
Fathoming rule - 2 is based on lower bound LB1(k) and is implemented by calling subroutine FATHM5.
Fathoming rule - 3 is based on lower bound LB2(k) and is implemented by calling subroutine FATHM3.
Fathoming rule - 4 is based on Corollary 5 and is implemented by calling subroutine REMJBP.

Four different programs are used for the experiment proposed in this dissertation. Programs 1-3 use Branch rule - 1 dicussed in Chapter 4, and Program 4 uses Branch rule - 2 also dicussed in Chapter 4.

Fathoming rule - 1 and fathoming rule - 4 have similar characteristics in the sense that they are derived from the same precedence matrix, and fathoming rule - 1 shows most of
the time a slightly better performance than fathoming rule-4, as is shown in Table 6.1.

Both fathoming rule-2 and fathoming rule-3 deal with the cost bound in the Branch and Bound method, and fathoming rule-2 is much stronger than fathoming rule-3, as is shown in Table 6.1. Therefore, the combination of fathoming rule-1 and fathoming rule-2 is considered to be a basic fathoming rule and is implemented in Programs 1-3. Branch rule-2 in conjunction with precedence matrix, described in Appendix F, only needs cost bound as a bounding criterion because Branch rule-2 already exploits the precedence matrix in itself. Since fathoming rule-2 is more often better than fathoming rule-3, as is shown in Table 6.1, Program 4 uses only fathoming rule-2 with the use of Branch rule-2.

The summary of four different programs is as follows:

Program 1: using Branch rule-1 with fathoming rule-1 and rule-2, in that order

Program 2: using Branch rule-1 with fathoming rule-1, rule-2 and rule-3, in that order

Program 3: using Branch rule-1 with fathoming rule-1, rule-2 and rule-4, in that order

Program 4: using Branch rule-2 with fathoming rule-2.

A flow chart of Programs 1-3 is shown in Figure 5.3 in the form of subroutine ACTL, and a description of variables in the program is shown in Appendix H.
Flow Chart of Subroutine ACTL

Initializations

Generation of the problem
discussed in sec. 5.2 chap. 5: generate the value of TIME(i) and WEIT(i) for all i

Determination of a Precedence Matrix based on Theorem 7

Revision of the previous precedence matrix such that it represents immediate successor(s) and predecessor(s), described in subsection 4.1.1 in chapter 4

Determination of solution from natural order of jobs, introduced in sec. 4.3 in chapter 4, use it as an upper bound.

From precedence matrix, determination of the jobs which will be prefixed based on Corollary 2. Those jobs are set to PREF(Job #) = 1
From precedence matrix, identification of the jobs which should be assigned on the earliest available machine; Those jobs are set to ISET(Job #) = -i.

Branch & Bound starts.
see Appendix (I).

Figure 5.3. Flow Chart of Subroutine ACTL
Chapter 6

RESULTS, CONCLUSIONS AND RECOMMENDATIONS

6.1 RESULTS AND CONCLUSIONS

This dissertation develops an algorithm of finding an optimal solution for the Weighted Flow Time Scheduling problem. In this chapter, computer times are reported for 15 different randomly generated problems introduced in Section 2 of Chapter 5 with various number of machines. Four different programs discussed in Section 3 of Chapter 5 and one heuristic, called H1, are used to solve the above problems.

A priority rule of heuristic H1 is implemented by assigning one job at a time in natural order to the earliest available machine until all jobs are assigned. The computational results are summarized in Table 6.1, and summaries of this section are as follows.

1. As is noted in Table 6.1, among the four programs, Program 2 takes the largest computer time to solve the same set of problems. The extreme case is that Program 2 takes usually 10 times more computer time than Program 1 for solving the same set of problems.
Table 6.1. Summarized Computational Results

<table>
<thead>
<tr>
<th>Problem #</th>
<th>no. of machines</th>
<th>Program 1</th>
<th>Program 2</th>
<th>Program 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no. of nodes</td>
<td>CPU</td>
<td>no. of nodes</td>
<td>CPU</td>
</tr>
<tr>
<td>1. 5 jobs</td>
<td>2</td>
<td>3</td>
<td>0.0005</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>0.0005</td>
<td>2</td>
</tr>
<tr>
<td>2. 7 jobs</td>
<td>2</td>
<td>16</td>
<td>0.0018</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>28</td>
<td>0.0026</td>
<td>27</td>
</tr>
<tr>
<td>3. 10 jobs</td>
<td>2</td>
<td>189</td>
<td>0.0155</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>658</td>
<td>0.054</td>
<td>655</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1054</td>
<td>0.091</td>
<td>1054</td>
</tr>
<tr>
<td>4. 13 jobs</td>
<td>2</td>
<td>851</td>
<td>0.060</td>
<td>851</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7203</td>
<td>0.55</td>
<td>7122</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>33840</td>
<td>2.89</td>
<td>33840</td>
</tr>
<tr>
<td>5. 15 jobs</td>
<td>2</td>
<td>1981</td>
<td>1.13</td>
<td>1981</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>29369</td>
<td>2.119</td>
<td>29209</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>233420</td>
<td>18.47</td>
<td>*</td>
</tr>
<tr>
<td>6. 17 jobs</td>
<td>2</td>
<td>6134</td>
<td>0.42</td>
<td>6134</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>149006</td>
<td>10.62</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1259725</td>
<td>98.78</td>
<td>*</td>
</tr>
<tr>
<td>7. 19 jobs</td>
<td>2</td>
<td>17827</td>
<td>1.25</td>
<td>17827</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>700035</td>
<td>51.51</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>8. 21 jobs</td>
<td>2</td>
<td>51126</td>
<td>3.5</td>
<td>51126</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>9. 23 jobs</td>
<td>2</td>
<td>207690</td>
<td>14.22</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>10. 25 jobs</td>
<td>2</td>
<td>414001</td>
<td>28.32</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>11. 26 jobs</td>
<td>2</td>
<td>630570</td>
<td>43.8</td>
<td>*</td>
</tr>
</tbody>
</table>

* Problem that can not be solved within three minutes of CPU
<table>
<thead>
<tr>
<th>Problem #</th>
<th>no. of machines</th>
<th>no. of nodes</th>
<th>CPU</th>
<th>optimal cost</th>
<th>cost from H1</th>
<th>%**</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>2</td>
<td>5</td>
<td>.00166</td>
<td>10187</td>
<td>10187</td>
<td>100 %</td>
</tr>
<tr>
<td>5 jobs</td>
<td>3</td>
<td>5</td>
<td>.00177</td>
<td>8846</td>
<td>8846</td>
<td>100 %</td>
</tr>
<tr>
<td>2.</td>
<td>2</td>
<td>16</td>
<td>.00629</td>
<td>17708</td>
<td>17723</td>
<td>99.92%</td>
</tr>
<tr>
<td>7 jobs</td>
<td>3</td>
<td>16</td>
<td>.1865</td>
<td>45625</td>
<td>45811</td>
<td>99.59%</td>
</tr>
<tr>
<td>3.</td>
<td>2</td>
<td>236</td>
<td>.177</td>
<td>60163</td>
<td>60163</td>
<td>100 %</td>
</tr>
<tr>
<td>10 jobs</td>
<td>3</td>
<td>524</td>
<td>.24</td>
<td>38730</td>
<td>38776</td>
<td>99.88%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>729</td>
<td>.24</td>
<td>38730</td>
<td>38776</td>
<td>99.88%</td>
</tr>
<tr>
<td>4.</td>
<td>2</td>
<td>1585</td>
<td>.57</td>
<td>90035</td>
<td>90049</td>
<td>99.98%</td>
</tr>
<tr>
<td>13 jobs</td>
<td>3</td>
<td>7756</td>
<td>2.73</td>
<td>66641</td>
<td>67108</td>
<td>99.30%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>25573</td>
<td>9.17</td>
<td>48666</td>
<td>48674</td>
<td>99.98%</td>
</tr>
<tr>
<td>5.</td>
<td>2</td>
<td>3413</td>
<td>1.325</td>
<td>107959</td>
<td>107959</td>
<td>100 %</td>
</tr>
<tr>
<td>15 jobs</td>
<td>3</td>
<td>35964</td>
<td>12.826</td>
<td>79221</td>
<td>79221</td>
<td>100 %</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>168466</td>
<td>59.33</td>
<td>56896</td>
<td>57276</td>
<td>99.33%</td>
</tr>
<tr>
<td>6.</td>
<td>2</td>
<td>11903</td>
<td>4.4</td>
<td>116799</td>
<td>116824</td>
<td>99.98%</td>
</tr>
<tr>
<td>17 jobs</td>
<td>3</td>
<td>189553</td>
<td>68.9</td>
<td>85268</td>
<td>85355</td>
<td>99.90%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>*</td>
<td></td>
<td>60703</td>
<td>61155</td>
<td>99.26%</td>
</tr>
<tr>
<td>7.</td>
<td>2</td>
<td>32965</td>
<td>13.39</td>
<td>158869</td>
<td>158887</td>
<td>99.99%</td>
</tr>
<tr>
<td>19 jobs</td>
<td>3</td>
<td>*</td>
<td>114709</td>
<td>114743</td>
<td>99.97%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>*</td>
<td></td>
<td>60703</td>
<td>61155</td>
<td>99.26%</td>
</tr>
<tr>
<td>8.</td>
<td>2</td>
<td>92903</td>
<td>38.18</td>
<td>180029</td>
<td>180122</td>
<td>99.95%</td>
</tr>
<tr>
<td>21 jobs</td>
<td>3</td>
<td>*</td>
<td></td>
<td>60703</td>
<td>61155</td>
<td>99.26%</td>
</tr>
<tr>
<td>9.</td>
<td>2</td>
<td>421233</td>
<td>174.55</td>
<td>193352</td>
<td>193402</td>
<td>99.97%</td>
</tr>
<tr>
<td>23 jobs</td>
<td>3</td>
<td>*</td>
<td></td>
<td>217895</td>
<td>218092</td>
<td>99.91%</td>
</tr>
<tr>
<td>10.</td>
<td>2</td>
<td>*</td>
<td></td>
<td>217895</td>
<td>218092</td>
<td>99.91%</td>
</tr>
<tr>
<td>25 jobs</td>
<td>3</td>
<td>*</td>
<td></td>
<td>232373</td>
<td>232382</td>
<td>99.99%</td>
</tr>
<tr>
<td>11.</td>
<td>2</td>
<td>*</td>
<td></td>
<td>232373</td>
<td>232382</td>
<td>99.99%</td>
</tr>
</tbody>
</table>

* Problem that can not be solved within three minutes of CPU.

** Percentage of \(1 - (\text{Heuristic sol.} - \text{optimal})/\text{optimal}\)
Table 6.1 (continued)

<table>
<thead>
<tr>
<th>Problem #</th>
<th>no. of machines</th>
<th>Program 1</th>
<th>Program 2</th>
<th>Program 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>no. of nodes</td>
<td>CPU</td>
<td>no. of nodes</td>
</tr>
<tr>
<td>12. 27 jobs</td>
<td>2</td>
<td>643877</td>
<td>44.69</td>
<td>623101</td>
</tr>
<tr>
<td>13. 28 jobs</td>
<td>2</td>
<td>1204664</td>
<td>82.62</td>
<td>1163250</td>
</tr>
<tr>
<td>14. 29 jobs</td>
<td>2</td>
<td>2179305</td>
<td>156.96</td>
<td>*</td>
</tr>
<tr>
<td>15. 30 jobs</td>
<td>2</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem #</th>
<th>no. of machines</th>
<th>Program 4</th>
<th>optimal cost</th>
<th>cost from H1</th>
<th>%**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>no. of nodes</td>
<td>CPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. 27 jobs</td>
<td>2</td>
<td>*</td>
<td></td>
<td>233030</td>
<td>233038</td>
</tr>
<tr>
<td>13. 28 jobs</td>
<td>2</td>
<td>*</td>
<td></td>
<td>260938</td>
<td>260942</td>
</tr>
<tr>
<td>14. 29 jobs</td>
<td>2</td>
<td>*</td>
<td></td>
<td>269895</td>
<td>289895</td>
</tr>
<tr>
<td>15. 30 jobs</td>
<td>2</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Problem that can not be solved within three minutes of CPU.

**Percentage of \[1 - (\text{Heuristic sol.} - \text{optimal})/\text{optimal}\]
2. Program 1 has fathoming rule-1 and fathoming rule-2, and Program 2 has fathoming rule-1, fathoming rule-2 and fathoming rule-3. Therefore, the only difference between Program 1 and Program 2 is that Program 2 has one additional fathoming rule, namely fathoming rule-3. Even though the number of nodes required to run Program 2 is a little less than the one required to run Program 1, it is too costly to add fathoming rule-3 over fathoming rule-1 and fathoming rule-2. From the observations made here, it is also shown that fathoming rule-2 (lower bound LB1) overpowers fathoming rule-3 (lower bound LB2) because the computer time needed to implement fathoming rule-3 is about 9 times more than the one needed to run the whole Program 1 which includes both fathoming rule-1 and fathoming rule-2.

3. Program 1, in most cases, is slightly better than Program 3. The only difference between Program 1 and Program 3 is that Program 3 has one more fathoming rule, namely fathoming rule-4. Since adding fathoming rule-4 over fathoming rule-1 and 2 is not costly, as is shown in Table 6.1, it is worth pursuing Program 3 in solving different types of problems. The only difference between Program 1 and Program 3 is that Problem #14 in Appendix J is solved by
Program 1 but not by Program 3 with the preset of a maximum of 3 minutes of total computer time.

4. Less than one second is required to solve problems of up to 17 jobs and 2 machines with Program 1 and Program 3; less than one minute is required to solve problems of up to 27 jobs and 2 machines with Program 1, indicating that such modest-sized problems can still be solved within a reasonable amount of computer time.

5. Program 4 is always worse than Program 1 and Program 3. As is shown in Section 2 of Chapter 4, the way branching is performed in Program 4 has nothing to do with the number of machines. Therefore, as the number of machines is increased, the performance of Program 4 is progressively better than that of Program 1. Nevertheless, the CPU time of Program 4 is at least 3 to five times larger than that of Program 1. The only difference between Program 1 and Program 4 is that Program 1 uses Branch rule-1, and Program 4 uses Branch rule-2. Therefore, Branch rule-1 overpowers Branch rule-2.

6. Heuristic H1 always generates its solution within 1% of optimal solutions. Therefore, when one is forced to solve large-sized problems with current
computing facilities, one may safely use heuristic H1 to generate a very close-to-optimum solution.

7. Solution time consistently increases as the problem size increases, indicating that solution procedures for the problems generated in this dissertation are very consistent with respect to problem size. If problems were not consistent in this way, solution time could either increase or decrease as problem size increases, and, as a result, comparison of different fathoming rules and branch rules could quite possibly be misleading.

8. In considering many different aspects of mathematical programming approaches for the Weighted Flow Time Scheduling problem described in Chapter 3, the author has found that the branch and bound method is the best approach to this problem given the current state of the art of mathematical programming methods.

6.2 RECOMMENDATIONS FOR FUTURE RESEARCH

1. One possible extension from this dissertation is to assume a general (say, piece-wise linear and convex, or general quadratic form) rather than a linear penalty function. Then, one would need to modify theorems on job ordering to some extent.
2. Another possible extension from this dissertation is to consider the case that jobs dynamically arrive at the system. Then one would expect that an optimal solution could result even if some machines experience idle time before all jobs are processed in the system. Then, one would need to modify Lemma 1.

3. If there is an à priori precedence relationship among jobs, much stronger fathoming rules could be developed. Then, one would need to modify fathoming rule-1 and fathoming rule-4 in conjunction with these à priori precedence relationships among jobs. This could possibly be a much stronger bounding criterion than the ones introduced in this dissertation.

4. Set-up and processing times which are dependent on the order in which jobs are scheduled pose a radically different problem from that discussed in this dissertation. Consequently, most of the theoretical results obtained in this dissertation would be rendered inapplicable to the new problem.

5. Finally, while interest in the problem of scheduling jobs on identical machines is important in its own right, non-identical machines with general available time of machines could be another extension of this dissertation.
BIBLIOGRAPHY


Appendix A

TRANSPORTATION FORMULATION
OF FLOW TIME SCHEDULING PROBLEM (PAGE 9)

Let $t_{ij}$ = processing time for job $i$ on machine $j$.

Observation: Let $S$ be some schedule. Suppose job $i$ is processed on machine $m_j$ and let there be exactly $k-1$ jobs which are scheduled after job $i$ on machine $m_j$. One can write $T(i)$ in such a way as to isolate all those terms which depend on $t_{ij}$. It is not difficult to see that the coefficient of $t_{ij}$ is $k$; that is, $\sum_{i=1}^{m} T(i)$ is the sum of $k \cdot t_{ij}$ and $n - 1$ other terms not containing $t_{ij}$. Example: if job $i$ runs last on $m_j$, then the coefficient of $t_{ij}$ is 1, if job $i$ runs next-to-last on $m_j$ then the coefficient of $t_{ij}$ is 2, etc.

This observation leads to the following related problem.

From the $m \times n$ matrix $(t_{ij})$ of processing time, we form the possible cost elements as $(t_{ij}, 2t_{ij}, \ldots, nt_{ij})$ depending on the position of job $i$ on machine $j$.

Where $k \cdot t_{ij} (1 \leq k \leq n)$ is the contribution of job $i$ on $m_j$ to $\sum_{i=1}^{n} T(i)$ when exactly $k - 1$ jobs are scheduled after
job \( i \) on \( m_j \). Therefore, the cost matrix \( Q \) is \( mn \times n \), and elements of \( Q \) are denoted by \( q_{ij} \) where \( 1 \leq i \leq mn \) and \( 1 \leq j \leq n \).

A set of \( n \) elements \( q_{i_1}^{n_1}, \ldots, q_{i_n}^{n_n} \) is called a feasible set if no two elements are in the same row; the cost of such a set is given by \( \sum_{r=1}^{n} q_{i_r}^{r} \). Then one must find a feasible set with the smallest possible cost. Let \( q_{i_1}^{n_1}, \ldots, q_{i_n}^{n_n} \) be a feasible set with the smallest cost; this set determines an optimal schedule.

Corollary: If \( i^*_r > m \), and no element from \( i^*_r - m \) is in the feasible set, then there exists an \( s \) (\( 1 \leq s \leq n \)) such that \( i^*_s = i^*_r - m \).

Proof: If \( i^*_r > m \) and no element from \( i^*_r - m \) is in the feasible set, we can simply replace \( q_{i_r}^{r} \) by \( q_{i_r-m}^{r} \) without increasing the cost since \( q_{i_r}^{r} > q_{i_r-m}^{r} \).

(In the degenerate case when entries in a column are zero, \( i^*_r \) is assumed to be as small as possible).

Q.E.D.

The numbers \( i^*_r \) can be written uniquely as \( i^*_r = (k_r - 1)m + \Delta_r \), where \( 1 \leq k_r \leq n \) and \( 1 \leq \Delta_r \leq m \), then in an optimal schedule, job \( r \) should be positioned at the \( k_r^{th} \) from the last to be run on machine \( \Delta_r \).

An example is given in Figure A.1. An optimal schedule is determined by the circled elements of \( Q \) according to the definitions just given for \( k_r \) and \( \Delta_r \).

Example: \( n = 5 \)
\( m = 3 \)
Then the optimal schedule is

machine 1: job 2
machine 2: job 3 + job 1
machine 3: job 4 + job 5

The problem of finding a minimal cost feasible set for a matrix Q can be formulated as a transportation problem.

The network is shown in Fig. A.2.
Figure A.2. Network Diagram of Transportation Formulation

where the arcs are labeled with pairs \( (x, y) \) with \( x \) as the arc capacity and \( y \) as the cost per unit flow, and \( F_{\text{max}} \), the maximum flow, is equal to \( n \). Then the problem here is finding a minimal cost flow pattern which achieves the maximum total \( F_{\text{max}} \) through the network.
Appendix B

PROOF OF THEOREM 1 (PAGE 17)

Proof: Consider a sequence $S$ that is not in natural order. That is, somewhere in $S$ there must exist a pair of adjacent jobs, $i$ and $j$, with $j$ following $i$, such that $\frac{t_i}{w_i} > \frac{t_j}{w_j}$. Now construct a new sequence, $S'$, in which job $i$ and $j$ are interchanged in sequence, and all other jobs are completed at the same time as in $S$. The situation is depicted in Fig. B.1.

Figure B.1. Diagram of Schedule $S$

where

- $A$ is a set of jobs preceding job $i$ and $j$
- $B$ is a set of jobs following job $i$ and $j$
- $t_a$ is the point in time at which job $i$ begins operation in $S$ and job $j$ begins operation in $S'$. 

80
Under schedule $S$, 

$$\text{Cost}(s) = \sum_{i=1}^{n} T(i) w_i = \sum_{k \in A} T(k) w_k + (t_a + t_i) w_i$$

$$+ (t_a + t_i + t_j) w_j + \sum_{k \in B} T(k) w_k$$

Under schedule $S'$, 

$$\text{Cost}(S') = \sum_{i=1}^{n} T(i) w_i = \sum_{k \in A} T(k) w_k + (t_a + t_j) w_j$$

$$+ (t_a + t_j + t_i) w_i + \sum_{k \in B} T(k) w_k$$

Therefore, 

$$\text{Cost}(S) - \text{Cost}(S') = (t_a + t_i) w_i + (t_a + t_i + t_j) w_j$$

$$- (t_a + t_j) w_j - (t_a + t_j + t_i) w_i$$

$$= t_i w_j - t_j w_i > 0$$

because $t_i/w_i > t_j/w_j$

Therefore, if the jobs $i$ and $j$ interchanged, total weighted flow time is reduced. So any sequence that is not in natural order can be improved by such an interchange of adjacent pair of jobs.
Appendix C

PROOF OF THEOREM 2

Before getting into the proof, let us introduce a definition to be used in the proof.

Def. If there exists a nonsingular matrix $P$ such that $PQ'P = Q'$, we say that $Q'$ is congruent to $Q$ and that $Q'$ is obtained from $Q$ by a congruence transformation, see [32].

Proof: If we take a congruence transformation of $Q$, we obtain $Q'$ such as $Q' = PQP$,

where

$$P = \begin{pmatrix}
\frac{1}{t_1} & 0 & \ldots & 0 \\
0 & \frac{1}{t_2} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \frac{1}{t_n}
\end{pmatrix}$$

and $P$ is a nonsingular square matrix.

Since the property of being positive definite is not affected by a congruence transformation, if one shows $Q'$ is positive definite, so is $Q$. 

82
Since all $t_i$ and $w_i$ are arranged such as

$$\frac{t_1}{w_1} \leq \frac{t_2}{w_2} \leq \frac{t_3}{w_3} \ldots \leq \frac{t_n}{w_n},$$

we can rearrange the above inequality as

$$\frac{w_1}{t_1} \geq \frac{w_2}{t_2} \geq \frac{w_3}{t_3} \ldots \geq \frac{w_n}{t_n}.$$  

And let $a_i = \frac{w_i}{t_i}$.
Then

\[
Q' = \begin{pmatrix}
\frac{a_1}{2} & \frac{a_2}{2} & \cdots & \frac{a_n}{2} \\
\frac{a_2}{2} & \frac{a_3}{2} & \cdots & \frac{a_n}{2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{a_n}{2} & \frac{a_n}{2} & \cdots & \frac{a_n}{2}
\end{pmatrix}
\]

where \( a_1 > a_2 > a_3 > \cdots > a_{n-1} > a_n > 0 \) (\( \forall i \)).

If all principal minors of \( Q \) are positive, then \( Q \) is positive definite. Let \( r_i \) be the \( i \)th principal minor. By using induction rule,

\[
\begin{align*}
i = 1, & \quad r_1 = |a_1| > 0 \\
i = 2, & \quad r_2 = \begin{vmatrix} a_1 & a_2/2 \\ a_2/2 & a_2 \end{vmatrix} = a_1a_2 - \frac{1}{4}a_2^2 > 0 \\
( & \because a_1 > a_2 > 0 )
\end{align*}
\]

\[
i = k, \text{ let us assume } r_k \text{ be positive for } k > 2
\]
then, for \( i = k + 1, \)
\[ r_{k+1} = \begin{bmatrix}
    a_1 & a_2 & a_3 & a_4 & a_k & a_{k+1} \\
    2 & 2 & 2 & 2 & 2 & 2 \\
    a_2 & a_2 & a_3 & a_4 & a_k & a_{k+1} \\
    2 & 2 & 2 & 2 & 2 & 2 \\
    a_3 & a_3 & a_4 & a_k & a_{k+1} \\
    2 & 2 & 2 & 2 & 2 & 2 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    a_{k+1} & a_{k+1} \\
    2 & 2 & 2 & 2 & \cdots & a_{k+1}
\end{bmatrix} \]

\[ = \begin{bmatrix}
    a_1 - \frac{a_2}{2} & -\frac{a_2}{2} & 0 & 0 & \cdots & 0 \\
    2 & 2 & 0 & 0 & \cdots & 0 \\
    a_2 & a_2 & a_3 & a_4 & \cdots & \frac{a_{k+1}}{2} \\
    2 & 2 & 2 & 2 & \cdots & \frac{2}{2} \\
    a_3 & a_3 & a_4 & a_k & \cdots & \frac{a_{k+1}}{2} \\
    2 & 2 & 2 & 2 & \cdots & \frac{2}{2} \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    a_{k+1} & a_{k+1} \\
    2 & 2 & 2 & 2 & \cdots & a_{k+1}
\end{bmatrix} \]
\begin{align*}
&= \left( a_1 - \frac{a_2}{2} \right) R_k + \frac{a_2}{2} R'_k \\
&\text{Now, by assumption,}

r_k &= \begin{array}{cccc}
    a_1 & a_2 & a_3 & \cdots & a_k \\
    2 & 2 & 2 & \cdots & 2 \\
\end{array} > 0
\end{align*}
Consider $R^*$. Note that the structure of $R^*$ is identical to that of $r^*$ and $a_1 \geq a_2 \ldots \geq a_k \geq a_{k+1} > 0$ by natural ordering of jobs. Hence, $R^*$ is positive since $r^*$ is positive by assumption.

Now, consider $R'_k$.

\[
R'_k = \begin{pmatrix}
\frac{1}{2}(a_2 - a_3) & -\frac{a_3}{2} & 0 & 0 & \cdots & 0 \\
\frac{a_3}{2} & a_3 & a_4 & a_5 & \cdots & a_{k+1} \\
\frac{a_4}{2} & a_3 & \frac{a_4}{2} & a_5 & \cdots & a_{k+1} \\
\frac{a_5}{2} & \frac{a_4}{2} & a_3 & \frac{a_4}{2} & \cdots & a_{k+1} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{a_k}{2} & \frac{a_{k-1}}{2} & \frac{a_k}{2} & \frac{a_{k-1}}{2} & \cdots & \frac{a_k}{2} \\
\frac{a_{k+1}}{2} & \frac{a_k}{2} & \frac{a_{k+1}}{2} & \frac{a_k}{2} & \cdots & \frac{a_{k+1}}{2}
\end{pmatrix}
\]

\[
= \frac{1}{2}(a_2 - a_3) \times \begin{pmatrix}
a_3 & a_4 & a_5 & \cdots & a_{k+1} \\
\frac{a_4}{2} & \frac{a_4}{2} & a_5 & \cdots & a_{k+1} \\
\frac{a_5}{2} & \frac{a_5}{2} & \frac{a_5}{2} & \cdots & a_{k+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{a_k}{2} & \frac{a_k}{2} & \frac{a_k}{2} & \cdots & \frac{a_k}{2} \\
\frac{a_{k+1}}{2} & \frac{a_{k+1}}{2} & \frac{a_{k+1}}{2} & \cdots & \frac{a_{k+1}}{2}
\end{pmatrix} + \frac{a_2}{2} \times \begin{pmatrix}
a_3 & a_4 & a_5 & \cdots & a_{k+1} \\
\frac{a_4}{2} & \frac{a_4}{2} & a_5 & \cdots & a_{k+1} \\
\frac{a_5}{2} & \frac{a_5}{2} & \frac{a_5}{2} & \cdots & a_{k+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{a_k}{2} & \frac{a_k}{2} & \frac{a_k}{2} & \cdots & \frac{a_k}{2} \\
\frac{a_{k+1}}{2} & \frac{a_{k+1}}{2} & \frac{a_{k+1}}{2} & \cdots & \frac{a_{k+1}}{2}
\end{pmatrix}
\]
\[ = \frac{1}{2}(a_2 - a_3)R_{k-1} + \frac{a_3}{2}R_{k-1} \]

Here again \( R_{k-1} > 0 \) with the same argument of \( R_k \) being positive.

Then we continue to decompose \( R'_1 \) for \( l = k-1, k-2, \ldots, 2, 1 \).

\[
R'_{k-1} = \frac{1}{2}(a_3 - a_4)R_{k-2} + \frac{a_4}{2}R_{k-2}
\]

\[
\vdots
\]

\[
R'_3 = \frac{1}{2}(a_{k-1} - a_k)R_2 + \frac{a_k}{2}R_2
\]

\[
R'_2 = \frac{1}{2}(a_k - a_{k+1})R_1 + \frac{a_{k+1}}{2}R_1
\]

\[
R'_1 = \frac{a_{k+1}}{2}
\]

Since \( R'_1 > 0 \Rightarrow R'_2 > 0 \Rightarrow \cdots \Rightarrow R'_k > 0 \)

Therefore, \( r_{k+1} > 0 \)

So, we can conclude that \( r_k \) is positive for all \( k \)

Q. E. D.
Proof: Suppose in a schedule $S$, job $j$ precedes job $i$ under the conditions of the Theorem 7. Since $t_i \leq t_j$ and $w_i \geq w_j$, implies $t_i/w_i \leq t_j/w_j$, if jobs $i$ and $j$ are on the same machine, one can easily reduce the total weighted flow time of the schedule $S$ by interchanging jobs $i$ and $j$ (see Appendix B). If these two jobs are on different machines, let job $i$ be on machine 1 and job $j$ be on machine 2 in the schedule $S$. This situation is depicted in Figure D.1.

\[\begin{array}{c|c|c|c}
\text{machine 1} & \text{Jobs in A} & \text{job i} & \text{Jobs in B} \\
\hline
& t_a & & \\
\text{machine 2} & \text{Jobs in A'} & \text{job j} & \text{Jobs in B'} \\
\hline
& t_b & & \\
\end{array}\]

Figure D.1. Diagram of Schedule $S$
where \( t_a \geq t_b \)

A is a set of jobs preceding job \( i \) on machine 1

B is a set of jobs following job \( i \) on machine 1

A' is a set of jobs preceding job \( j \) on machine 2

B' is a set of jobs following job \( j \) on machine 2

Then,

\[
\text{Cost}(S) = \sum_{k \in E} T(k)w_k + (t_a + t_i)w_i + (t_b + t_j)w_j
\]

where \( E = A \cup A' \cup B \cup B' \)

Let \( W_1 = \sum_{i \in B} w_i \) and \( W_2 = \sum_{i \in B'} w_i \)

and \( \overline{B} = B \cup \{i\}, \overline{B'} = B' \cup \{j\} \)

i) If \( W_1 \geq W_2 \), consider a schedule \( S' \) by interchanging jobs in \( \overline{B} \) and jobs in \( \overline{B'} \) from schedule \( S \). Schedule \( S' \) is shown in Fig. D.2.

![Figure D.2. Diagram of Schedule S'](image)

Then,

\[
\text{Cost}(S') = \sum_{k \in E} T(k)w_k + (t_a + t_i)w_i + (t_b + t_j)w_j
\]
So, \( \text{Cost}(S) - \text{Cost}(S') = (t_a - t_b)(W_i - W_j + w_i - w_j) \geq 0 \)

ii) If \( W_1 < W_2 \), let's consider a schedule \( S'' \) by interchanging job \( i \) and \( j \) from schedule \( S \). Schedule \( S'' \) is shown in Fig. D.3.

Figure D.3. Diagram of Schedule \( S'' \)

Then,
\[
\text{Cost}(S'') = \sum_{k \in E} T(k) \cdot w_k + (t_a + t_j)w_j + (t_b + t_i)w_i
\]

So, \( \text{Cost}(S) - \text{Cost}(S'') = (t_a - t_b)(W_i - W_j) + (t_j - t_i)(W_2 - W_1) \geq 0 \)

Therefore, there exists another schedule (\( S' \) or \( S'' \), depending on the conditions of \( W_1 \geq W_2 \) or \( W_1 < W_2 \)) where job \( i \) precedes job \( j \), for which case total cost is either decreased or left unchanged. So any schedule in which job \( j \) precedes job \( i \) can be improved by such an interchange of either jobs \( i \) and \( j \) or the string of jobs following job \( i \) and job \( i \) itself, and the string of jobs following job \( j \) and job \( j \) itself, depending on whether \( W_1 \geq W_2 \) or \( W_1 < W_2 \).
Appendix E

PROOF OF EQUATION (4.1) (PAGE 39)

Let \( r_i = \frac{w_i}{t_i} \). Then \( r_i \leq r_j \) for all jobs \( i > j \),
to keep in natural order. The proof of Eq. (4.1) is completed by induction rule.

Lemma 1: If \( r_i = \mathcal{A} \) for \( i = 1, 2, \ldots, n \), then

\[
C_{1,n} = \frac{1}{2} \mathcal{A} \left[ (\sum_{i} t_i)^2 + \sum_{i} t_i^2 \right] \quad \ldots \ldots \ldots \quad \text{(E.1)}
\]

Proof: \( C_{1,n} = \sum_{j} \sum_{i \leq j} w_j t_i \)

\[
= \mathcal{A} \sum_{j} \sum_{i \leq j} t_j t_i
\]

\[
= \mathcal{A} \left[ \sum_{i,j} t_i t_j + \sum_{i} t_i^2 \right]
\]

But \( \sum_{i,j} t_i t_j = \sum_{i} t_i \cdot \sum_{j} t_j = (\sum_{i} t_i)^2 \)

Therefore, \( C_{1,n} = \frac{1}{2} \mathcal{A} \left[ (\sum_{i} t_i)^2 + \sum_{i} t_i^2 \right] \)

Proposition: If \( r_i = \mathcal{A} \) for \( i = 1, 2, \ldots, n \), then the
Eq. (4.1) holds.
Proof: Let \( \Sigma_k( ) \) denote summation of the quantity inside the parentheses over all jobs assigned to machine \( k(k = 1, \ldots, m) \) in the given schedule \( S \). Let \( U_k = \Sigma_k t_i \) be the total time consumed on machine \( k \). Then by eq. (E.1), the penalty cost for machine \( k \) is
\[
\frac{1}{2} \alpha \left[ \left( \Sigma_k t_i \right)^2 + \Sigma_k t_i^2 \right]
\]
Consequently, the penalty cost of the schedule \( S \) is
\[
C_{m,n} = \frac{1}{2} \alpha \sum_{k=1}^{m} \left( U_k^2 + \Sigma_k t_i^2 \right)
\]
\[
= \frac{1}{2} \alpha \sum_{k=1}^{m} U_k^2 + \frac{1}{2} \sum_{k=1}^{m} \Sigma_k (\alpha t_i) t_i.
\]
But \( \alpha t_i = w_i \) and \( \sum_{i=1}^{n} w_i t_i = C_{n,n} \). Therefore
\[
C_{m,n} = \frac{1}{2} \alpha \sum_{k=1}^{m} U_k^2 + \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{m} (\alpha t_i) t_i \quad \text{(E.2)}
\]
and \( (\sum_i x_i)^2 \leq n \sum x_i^2 \quad \text{(E.3)} \)
(E.3) is the special case of Cauchy's inequality. Thus, from eq. (E.2) and eq. (E.3)
\[
C_{m,n} - \frac{1}{2} C_{n,n} = \frac{1}{2} \alpha \sum_{k=1}^{m} U_k^2 \geq \frac{1}{2m} \alpha (\sum_{k=1}^{n} U_k)^2
\]
From eq. (E.1),

\[ c_{1,n} = \frac{1}{2} \alpha (\sum t_i)^2 + \frac{1}{2} c_{n,n} \quad (E.5) \]

So, \[ c_{1,n} = \frac{1}{2} c_{n,n} = \frac{1}{2} \alpha (\sum t_i)^2 = \frac{1}{2} \left( \sum_{k=1}^{m} u_k \right)^2 \]

-------------------- (E.6)

From eq. (E.4) and eq. (E.6)

\[ c_{m,n} - \frac{1}{2} c_{n,n} \geq \frac{1}{2m} \alpha (\sum_{k=1}^{n} u_k)^2 = \frac{1}{m} (c_{1,n} - \frac{1}{2} c_{n,n}) \]

Q.E.D.

The proof of eq. (4.1) proceeds by induction on the number of distinct values of the ratios \( r_i \). The case \( \gamma = 1 \) has been proved. Assume that eq. (4.1) holds for every schedule when \( \gamma = k - 1 \), and consider the case \( \gamma = k \).

Let \( q \) be the number of jobs whose ratios \( r_i \) are equal to \( r_i \), and define

\[ z = (r_{q+1})/r_q < 1 \]

Define a new collection of \( n \) jobs by revising the weight coefficients \( w_i \) of the original jobs as follows: Let

\[ w'_i = z w_i \text{ for } 1 \leq i \leq q \]
\[ = w_i \text{ for } q + 1 \leq i \leq n, \]

where the prime symbol is used to distinguish the new jobs from the original jobs.
We now have

\[ r'_1 = r'_2 = \ldots = r'_q = r'_{q+1} = r_{q+1} \]
\[ r'_{q+2} = r_{q+2} \]
\[ r'_n = r_n. \]

so that the number of distinct ratios for the new set of jobs is \( k - 1 \). Applying the inductive hypothesis,

\[ C'_{m,n} - \frac{1}{2} C'_{n,n} \geq (1/m) \left( C'_{1,n} - \frac{1}{2} C'_{n,n} \right) \]

for any scheduling of new jobs, and in particular for schedule \( S \). Let

\[ U = C'_{1,n} - \frac{1}{2} C'_{n,n}, \quad V = C'_{m,n} - \frac{1}{2} C'_{n,n} \]
\[ U' = C'_{1,n} - \frac{1}{2} C'_{n,n}, \quad V' = C'_{m,n} - \frac{1}{2} C'_{n,n} \]

Then

\[ U' \leq m V' \quad (E.7) \]

Let us divide each of the quantities \( U' \) and \( V' \) into two parts,

\[ U' = U'' + U''' \]
\[ V' = V'' + V''' \]

where \( U'' \) is the part of \( U' \) comprised of penalty costs of the first \( q \) jobs, and \( U''' \) is the part comprised of penalty costs of the remaining \( n - q \) jobs of the new set; and similarly
for $V''$ and $V'''$. Since the first $q$ jobs have equal ratios, it follows from the Proposition that

$$U'' \leq m V'' \quad (E.8)$$

If the new jobs are assigned by schedule $S$, then we have

$$U = U'' + \left(\frac{1}{z}\right) U'' \quad \text{and} \quad V = V''' + \left(\frac{1}{z}\right) V'' \quad (E.9)$$

If $U'' \leq m V'''$, then by eq. (E.8) and eq. (E.9)

$$U \leq m V$$

Completing the proof.

Otherwise, we have

$$U'' > m V''' \quad (E.10)$$

Since for any collection of jobs and any schedule $U > 0$ and $V > 0$, we have from eq. (E.8) and eq. (E.10)

$$m V'' U'' > m V''' U''$$

and hence,

$$V'' U'' > V''' U''$$

and $(1-z) V'' U'' > (1-z) V''' U''$

Therefore, it follows that

$$z U' V = (U'' + U''')(z V''' + V'') > (V'' + V''')(z U'' + U'') = z V' U$$

Thus

$$U' V > V' U,$$

and by eq. (E.7)

$$m V' V \geq U' V \geq V' U$$

or again $m V \geq U$

Completing the proof for the case $\nu = k$.

Q.E.D.
Appendix F

BRANCH RULE - 2 IN CONJUNCTION WITH PRECEDENCE MATRIX

Let $P(k)$ be a set of $k$ jobs already assigned to machines.

- $F(i)$ be the set of immediately succeeding jobs of $i$ described in subsec. 4.1.1.
- $B(i)$ be the set of immediately preceding jobs of $i$ described in sec. 1 of Chapter 4.
- $L(i)$ be a set of jobs of which immediate ascendant is the same as the immediate ascendant of job $i$ in the tree. Here $L(1) = \emptyset$ is assumed, based on Corollary 1.

Then consecutive generation of $P(k)$ can be done by generating the immediate descendants of job $i$ in $P(k)$.

Let's suppose Fig. F.1 shows the immediate precedence relationships among jobs based on Theorem 7.
Figure F.1. Diagram of Precedence Relationships Among Jobs

where each node represents each job

Fig. F.1 shows that job 2 and 3 are immediate successors of job 1, and job 2 is immediate predecessor of job 4 and job 5 and etc. Then the search tree is shown in Fig. F.2.

Figure F.2. Diagram of Search Tree

At level 1, only job 1 exists, and P(1) = 1. Since L(1) = 0 and F'(1) = \{2, 3\}, the immediate descendants of job 1 is the job in set \{L(1) \cup F'(1)\} = \{2, 3\}. Therefore, at level 2, job 2 and job 3 lead the search tree. At node 2 in level 2, P(2) = \{1, 2\}, L(2) = \{3\} and F'(2) = \{4, 5\},
here job 5 is eliminated from set $F'(2)$ because job 3, which immediately precedes job 5 in Fig. F.1, is not assigned yet, i.e. $3 \notin P(2)$. Therefore the immediate descendants of job 2 in level 2, the jobs in set $\{L(2) \cup F'(2)\} = \{3, 4\}$. So, job 3 and job 4 will lead the subtree at level 3 generated from job 2 at level 2, and so on. Flow chart of Branch rule - 2 is shown in Appendix G.
Flow Chart of Program 4

Flow Chart

- Generation of the Problems, described in sec. 5.2 chap. 5.
- Find upper bound, based on in sec. 4.3, chapter 4
  - \( k = 1 \)
  - \( N(1) = 1 \)
  - \( JOB = 1 \)
- Generation of job i for next branch based on set \( \{L(JOB) \cup F'(JOB)\} \)
  - \( k = k + 1 \)
update necessary variables

update solution if current solution is less than ICUMB

Call FATHM5, check fathoming

N(k) = (N(k - 1)) \cup \{i\}

Figure G.1. Flow Chart of Program 4
Appendix H

DESCRIPTION OF VARIABLES IN COMPUTER PROGRAM

Variables:

TIME(i): processing time of job i
WEIT(i): weight of job i
PREF(i) = 1: job i should be assigned to the machine i at the optimal solution
ISET(i) = -i: job i should be assigned to the earliest available machine
ISTJB(i): starting time of job i
ICTMA(i): completion time of machine i
MACUJ(i) = k: job i is currently assigned to machine k
M(i, 1): number jobs currently assigned to machine i
M(i, j): job number assigned to machine i at (j -1)th position
ICCST: cost with current partial assignment
ICUMB: value of incumbent solution
ISOL(k): incumbent solution, where k has the size of (n + m)
IPRM(i, j) = 1: job i precedes job j, j > i
= 0: otherwise

102
IPRM(i, 1): number of jobs to precede job i
IPRM(i, j): job number to precede job i, \(2 \leq j \leq i\)
Appendix I

FLOW CHART OF BRANCH AND BOUND

I.1 FLOW CHART OF PROGRAMS 1-3

Flow chart of Programs 1-3

set
i = 1

IS
PREF(i) = 1

no
ISTOP = i

yes

i = i + 1

Assign job i to machine i and update necessary variables such as ISTJB(i), ICTMA(i), M(i, 1), M(i, 1), and ICCST.
Determine the earliest available machine, and let it be k. Then MACUJ(i) = k and update necessary variable.

Assign a job i to machine 1 and set MACUJ(i) = 1.

Check Fathomings, detailed Fig. I.2.1-4. Based on different programs discussed in sec. 5.3, chapter chapter 5.

i = i + 1

yes

no

i = n

yes

no
Backtrack

E
\[ i = i - 1 \]
\[ IS \]
\[ MACUJ(i) = m \]
\[ no \]
\[ IS \]
\[ ISET(i) = -1 \]
\[ no \]

move job \( i \) to the next machine:
\[ k = MACUJ(i) \]
\[ MACUJ(i) = k + 1 \]

update necessary variables

F
update solution if current solution is less than incumbent solution

Figure I.1.1. Flow Chart of Programs 1-3
I.2 FLOW CHART OF FOUR FATHOMING RULES

Subroutine FATHM 2

Let i be job # and j be machine # to the current assignment

\[ KK = IPRM(i, 1) \]
\[ KK1 = KK + 1 \]

\[ DO \ K = 2, KK1 \]

\[ JOB1 = IPRM(i, K) \]

**IS**

\[ ISTJB(i) \]
\[ ISTJB(JOB1) \]

Yes \[ \text{fathom} \]

No **IS**

\[ K = KK1 \]

Yes \[ \text{RETURN} \]

No \[ \text{No fathom} \]

Figure I.2.1. Flow Chart of Subroutine FATHM 2
Subroutine FATHM 5

Calculate LB1(i)
based on subsec. 4.1.2
of Chapter 4.

IS
LB1(i) > ICUMB

yes
fathom

no

No fathom

RETURN

Figure I.2.2. Flow Chart of Subroutine FATHM 5
Subroutine FATHM 3

With current partial assignment, identification of interaction terms among jobs, which should or should not included in ICCST

Calculate LB2(i) based on subsec. 4.1.3 of Chapter 4

\[ \text{LB2}(i) > \text{ICUMB} \]

- **yes** fathom
- **no** No fathom

Figure I.2.3. Flow Chart of Subroutine FATHM 3
Subroutine REMJBP

Identification of Job1 such that IPRM(i, Job1) = 0

where Job1 i

\[ KK = IPRM(Job1, 1) \]
\[ KK1 = KK + 1 \]

\[ T^* = \min_{i \in M} ICTMA(i) \]

DO K = 2, KK1

\[ JOB2 = IPRM(i, K) \]

IS

ISTJB(JOB2) \[ T^* \]

yes fathom

no

IS

K = KK1

yes No fathom

no

RETURN

Figure I.2.4. Flow Chart of Subroutine REMJBP
Appendix J

TABLE OF 15 PROBLEMS

Table J.1. List of 15 Problems

1. Problem 1 (5 Jobs)

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>23</td>
<td>34</td>
<td>57</td>
<td>82</td>
</tr>
<tr>
<td>$w_i$</td>
<td>18</td>
<td>79</td>
<td>45</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>0.16</td>
<td>0.29</td>
<td>0.75</td>
<td>1.4</td>
<td>2.8</td>
</tr>
</tbody>
</table>

2. Problem 2 (7 Jobs)

<table>
<thead>
<tr>
<th>Job</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>23</td>
<td>34</td>
<td>67</td>
<td>57</td>
<td>70</td>
<td>82</td>
</tr>
<tr>
<td>$w_i$</td>
<td>18</td>
<td>79</td>
<td>45</td>
<td>54</td>
<td>40</td>
<td>33</td>
<td>29</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>0.16</td>
<td>0.29</td>
<td>0.75</td>
<td>1.2</td>
<td>1.4</td>
<td>2.1</td>
<td>2.8</td>
</tr>
</tbody>
</table>

3. Problem 3 (10 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>23</td>
<td>65</td>
<td>34</td>
<td>61</td>
<td>89</td>
<td>57</td>
<td>70</td>
<td>87</td>
<td>82</td>
</tr>
<tr>
<td>$w_i$</td>
<td>18</td>
<td>79</td>
<td>96</td>
<td>45</td>
<td>54</td>
<td>65</td>
<td>40</td>
<td>33</td>
<td>35</td>
<td>29</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>0.16</td>
<td>0.29</td>
<td>0.67</td>
<td>0.75</td>
<td>1.1</td>
<td>1.3</td>
<td>1.4</td>
<td>2.1</td>
<td>2.4</td>
<td>2.8</td>
</tr>
</tbody>
</table>

4. Problem 4 (13 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>23</td>
<td>65</td>
<td>34</td>
<td>61</td>
<td>46</td>
<td>89</td>
<td>57</td>
<td>70</td>
<td>82</td>
<td>112</td>
<td>115</td>
</tr>
<tr>
<td>$w_i$</td>
<td>18</td>
<td>79</td>
<td>96</td>
<td>45</td>
<td>54</td>
<td>36</td>
<td>65</td>
<td>40</td>
<td>33</td>
<td>35</td>
<td>29</td>
<td>27</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>0.16</td>
<td>0.29</td>
<td>0.67</td>
<td>0.75</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>2.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Problem 5 (15 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>3</td>
<td>11</td>
<td>23</td>
<td>65</td>
<td>34</td>
<td>61</td>
<td>78</td>
<td>46</td>
<td>89</td>
<td>57</td>
</tr>
<tr>
<td>$w_i$</td>
<td>18</td>
<td>54</td>
<td>79</td>
<td>96</td>
<td>45</td>
<td>54</td>
<td>66</td>
<td>36</td>
<td>65</td>
<td>40</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>.16</td>
<td>.20</td>
<td>.29</td>
<td>.67</td>
<td>.75</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
</tr>
</tbody>
</table>

6. Problem 6 (17 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>2</td>
<td>3</td>
<td>11</td>
<td>23</td>
<td>23</td>
<td>65</td>
<td>34</td>
<td>61</td>
<td>78</td>
<td>46</td>
</tr>
<tr>
<td>$w_i$</td>
<td>48</td>
<td>18</td>
<td>54</td>
<td>79</td>
<td>51</td>
<td>96</td>
<td>45</td>
<td>54</td>
<td>66</td>
<td>36</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>.04</td>
<td>.16</td>
<td>.20</td>
<td>.29</td>
<td>.45</td>
<td>.67</td>
<td>.75</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
</tr>
</tbody>
</table>

7. Problem 7 (19 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>2</td>
<td>3</td>
<td>11</td>
<td>23</td>
<td>23</td>
<td>65</td>
<td>45</td>
<td>34</td>
<td>69</td>
<td>61</td>
</tr>
<tr>
<td>$w_i$</td>
<td>48</td>
<td>18</td>
<td>54</td>
<td>79</td>
<td>51</td>
<td>96</td>
<td>66</td>
<td>45</td>
<td>78</td>
<td>54</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>.04</td>
<td>.16</td>
<td>.20</td>
<td>.29</td>
<td>.45</td>
<td>.67</td>
<td>.68</td>
<td>.75</td>
<td>.88</td>
<td>1.1</td>
</tr>
</tbody>
</table>
### 8. Problem 8 (21 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>23</td>
<td>23</td>
<td>65</td>
<td>45</td>
<td>34</td>
<td>69</td>
</tr>
<tr>
<td>$w_x$</td>
<td>48</td>
<td>46</td>
<td>18</td>
<td>54</td>
<td>79</td>
<td>51</td>
<td>96</td>
<td>66</td>
<td>45</td>
<td>78</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>0.04</td>
<td>0.15</td>
<td>0.16</td>
<td>0.20</td>
<td>0.29</td>
<td>0.45</td>
<td>0.67</td>
<td>0.68</td>
<td>0.75</td>
<td>0.88</td>
</tr>
</tbody>
</table>

### 9. Problem 9 (23 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>23</td>
<td>24</td>
<td>65</td>
<td>45</td>
<td>34</td>
</tr>
<tr>
<td>$w_x$</td>
<td>48</td>
<td>46</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>79</td>
<td>51</td>
<td>51</td>
<td>96</td>
<td>66</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>0.04</td>
<td>0.15</td>
<td>0.16</td>
<td>0.20</td>
<td>0.21</td>
<td>0.29</td>
<td>0.45</td>
<td>0.67</td>
<td>0.68</td>
<td>0.75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>69</td>
<td>19</td>
<td>61</td>
<td>78</td>
<td>46</td>
<td>89</td>
<td>57</td>
<td>70</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>$w_x$</td>
<td>78</td>
<td>19</td>
<td>54</td>
<td>66</td>
<td>36</td>
<td>65</td>
<td>40</td>
<td>33</td>
<td>35</td>
<td>34</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>0.88</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>2.1</td>
<td>2.5</td>
<td>2.6</td>
</tr>
</tbody>
</table>
10. Problem 10 (25 Jobs)

<table>
<thead>
<tr>
<th>Job</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>87</td>
<td>82</td>
<td>20</td>
</tr>
<tr>
<td>$w_x$</td>
<td>33</td>
<td>29</td>
<td>4</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>2.6</td>
<td>2.8</td>
<td>5.0</td>
</tr>
</tbody>
</table>

11. Problem 11 (26 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>23</td>
<td>23</td>
<td>42</td>
<td>65</td>
</tr>
<tr>
<td>$w_x$</td>
<td>48</td>
<td>95</td>
<td>46</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>79</td>
<td>51</td>
<td>81</td>
<td>96</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>.04</td>
<td>.05</td>
<td>.15</td>
<td>.16</td>
<td>.20</td>
<td>.21</td>
<td>.29</td>
<td>.45</td>
<td>.51</td>
<td>.67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>45</td>
<td>34</td>
<td>69</td>
<td>19</td>
<td>61</td>
<td>78</td>
<td>46</td>
<td>89</td>
<td>57</td>
<td>70</td>
</tr>
<tr>
<td>$w_x$</td>
<td>66</td>
<td>45</td>
<td>78</td>
<td>19</td>
<td>54</td>
<td>66</td>
<td>36</td>
<td>65</td>
<td>40</td>
<td>33</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>.68</td>
<td>.75</td>
<td>.88</td>
<td>1.0</td>
<td>1.1</td>
<td>1.1</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>2.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>87</td>
<td>88</td>
<td>87</td>
<td>82</td>
<td>20</td>
</tr>
<tr>
<td>$w_x$</td>
<td>35</td>
<td>34</td>
<td>33</td>
<td>29</td>
<td>4</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>2.5</td>
<td>2.6</td>
<td>2.6</td>
<td>2.8</td>
<td>5.0</td>
</tr>
</tbody>
</table>
### 12. Problem 12 (27 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_\alpha$</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>42</td>
</tr>
<tr>
<td>$w_\alpha$</td>
<td>48</td>
<td>95</td>
<td>46</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>85</td>
<td>79</td>
<td>51</td>
<td>81</td>
</tr>
<tr>
<td>$t_\alpha/w_\alpha$</td>
<td>.04</td>
<td>.05</td>
<td>.15</td>
<td>.16</td>
<td>.20</td>
<td>.21</td>
<td>.28</td>
<td>.29</td>
<td>.45</td>
<td>.51</td>
</tr>
</tbody>
</table>

### 13. Problem 13 (28 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_\alpha$</td>
<td>70</td>
<td>87</td>
<td>88</td>
<td>87</td>
<td>82</td>
<td>20</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>85</td>
</tr>
<tr>
<td>$w_\alpha$</td>
<td>46</td>
<td>89</td>
<td>51</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_\alpha/w_\alpha$</td>
<td>.04</td>
<td>.05</td>
<td>.15</td>
<td>.16</td>
<td>.20</td>
<td>.21</td>
<td>.28</td>
<td>.29</td>
<td>.45</td>
<td>.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_\alpha$</td>
<td>65</td>
<td>45</td>
<td>34</td>
<td>69</td>
<td>19</td>
<td>61</td>
<td>78</td>
<td>46</td>
<td>89</td>
<td>57</td>
</tr>
<tr>
<td>$w_\alpha$</td>
<td>96</td>
<td>66</td>
<td>45</td>
<td>78</td>
<td>19</td>
<td>54</td>
<td>66</td>
<td>36</td>
<td>65</td>
<td>40</td>
</tr>
<tr>
<td>$t_\alpha/w_\alpha$</td>
<td>.67</td>
<td>.68</td>
<td>.75</td>
<td>.88</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_\alpha$</td>
<td>70</td>
<td>87</td>
<td>88</td>
<td>87</td>
<td>82</td>
<td>20</td>
<td>95</td>
</tr>
<tr>
<td>$w_\alpha$</td>
<td>33</td>
<td>35</td>
<td>34</td>
<td>33</td>
<td>29</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$t_\alpha/w_\alpha$</td>
<td>2.1</td>
<td>2.5</td>
<td>2.6</td>
<td>2.6</td>
<td>2.8</td>
<td>5.0</td>
<td>95</td>
</tr>
</tbody>
</table>
14. Problem 14 (29 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>42</td>
</tr>
<tr>
<td>$w_x$</td>
<td>48</td>
<td>95</td>
<td>46</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>85</td>
<td>79</td>
<td>51</td>
<td>81</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>.04</td>
<td>.05</td>
<td>.15</td>
<td>.16</td>
<td>.20</td>
<td>.21</td>
<td>.28</td>
<td>.29</td>
<td>.45</td>
<td>.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>50</td>
<td>65</td>
<td>45</td>
<td>34</td>
<td>54</td>
<td>69</td>
<td>19</td>
<td>61</td>
<td>78</td>
<td>46</td>
</tr>
<tr>
<td>$w_x$</td>
<td>84</td>
<td>96</td>
<td>66</td>
<td>45</td>
<td>69</td>
<td>78</td>
<td>54</td>
<td>66</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>.59</td>
<td>.67</td>
<td>.68</td>
<td>.75</td>
<td>.78</td>
<td>.88</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
</tr>
</tbody>
</table>

15. Problem 15 (30 Jobs)

<table>
<thead>
<tr>
<th>Job</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>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>11</td>
<td>12</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>42</td>
</tr>
<tr>
<td>$w_x$</td>
<td>48</td>
<td>95</td>
<td>46</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>85</td>
<td>79</td>
<td>51</td>
<td>81</td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>.04</td>
<td>.05</td>
<td>.15</td>
<td>.16</td>
<td>.20</td>
<td>.21</td>
<td>.28</td>
<td>.29</td>
<td>.45</td>
<td>.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_x$</td>
<td>50</td>
<td>65</td>
<td>45</td>
<td>34</td>
<td>54</td>
<td>69</td>
<td>19</td>
<td>61</td>
<td>78</td>
<td>46</td>
</tr>
<tr>
<td>$w_x$</td>
<td>84</td>
<td>96</td>
<td>66</td>
<td>45</td>
<td>69</td>
<td>78</td>
<td>54</td>
<td>66</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>$t_x/w_x$</td>
<td>.59</td>
<td>.67</td>
<td>.68</td>
<td>.75</td>
<td>.76</td>
<td>.78</td>
<td>.88</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
</tr>
<tr>
<td>Job</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>$t_i$</td>
<td>46</td>
<td>89</td>
<td>57</td>
<td>70</td>
<td>87</td>
<td>88</td>
<td>87</td>
<td>82</td>
<td>20</td>
<td>95</td>
</tr>
<tr>
<td>$w_i$</td>
<td>36</td>
<td>65</td>
<td>40</td>
<td>33</td>
<td>35</td>
<td>34</td>
<td>33</td>
<td>29</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$t_i/w_i$</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>2.1</td>
<td>2.5</td>
<td>2.6</td>
<td>2.6</td>
<td>2.8</td>
<td>5.0</td>
<td>95</td>
</tr>
</tbody>
</table>
Appendix K

LIST OF COMPUTER PROGRAMS

K.1 LIST OF PROGRAM 1

C. N: # OF JCB
C. ITLNM: TIPLE # OF GF MACHINES
CALL SUB(TIME,HEIT,RATIC,JBCHB,IRANK,IPRM,ISET,ISCL,
+ MACUJ,JBSh,STJB,ICTMA)
STOP
END
SUBROUTINE SUB(TIME,HEIT,RATIC,JBCHB,IRANK,IPRM,ISET,ISCL,
+ MACUJ,JBSh,STJB,ICTMA)
COMMON N,ITLNM,INCl,INCl2
DIMENSION TIME(N),HEIT(N),RATIC(N),JBCHB(N)
INTEGER*4 IRANK(N),IPRM(N,N),ISET(N),M(11KLM,1INC1)
+ IINC2,MACUJ(N),JBSh(N),STJB(N),ICTMA(ITLNM)
N=25
ITLNM=3
INCl=N+2
INCl2=N*ITLNM
N1=N
N2=N
N3=N
N4=N
N5=(N+1)/2
N6=(N*N+1)/2
N7=N5
N8=(ITLNM*INCl+1)/2
N9=(INCl2+1)/2
N10=N5
N11=N5
N12=N5
N13=(ITLNM+1)/2
CALL GETM(TIME,N1,HEIT,N2,RATIC,N3,JBCHB,N4,IRANK,N5,
+ IPRM,N6,ISET,N7,M,N8,ISCL,N5,MACUJ,NIC,JBSh,N11,
+ STJB,N12,ICTMA,N13)
CALL ACTL(TIME,WEIT,RATIC,JLBCB,IRANK,IPRM,ISET,M,  
+ ISCL,MACUJ,ISTJ8,ICTMA)
CALL FREEM(TIME,N1,WEIT,N2,RATIC,N3,JLBCB,N4,IRANK,N5  
+ IPRM,N6,ISET,N7,M,N8,ISCL,N9,MACUJ,N10,JLBSH,N11,  
+ ISTJB,N12,ICTMA,N13)
RETURN
END
SUBROUTINE ACTL(TIME,WEIT,RATIC,JLBCB,IRANK,IPRM,ISET,  
+ M,ISCL,MACUJ,ISTJ8,ICTMA)
COMMON N,ITLNM,IUNC1,IUNC2
DIMENSION TIME(N),WEIT(N),RATIC(N),JLBCB(N)
INTEGER*2 IRANKIN),IPRMIN,N),ISETINC)
  ,I  SOL(INCZ J,MACUj!N) ,JL8SK(N),ISTJB(IN),ICTMA(IN)
C
C NMFTS: # OF JOBS IN 1-ST SET IN S1, S1
C ICUMB: COST FOR A INCUMBENT SOLUTION
DO 1 I=1,N
  2 K1=K2
  CALL KERAND(K1,K2,F)
  IZ=F*IC**2)
  IF(IIZ.EC.G) GC TC 2
  3 K1=K2
  CALL KERAND(K1,K2,FJ
  IK=F*IC**2)
  IF(IK.EC.O) GC TC 3
  TIME(I)=IZ
  1 WEIT(I)=IK
C1; START: RANKING THE JOBS WITH RESPECT TO NONDECREASING CRITERION T  
C
C RATIC(1)=TIME(1)/WEIT(1)
  I=1
  IRANK(1)=1
  DO 40 I=2,N
  RATIC(I)=TIME(I)/WEIT(I)
  WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
  40 WRITE(6,52)
  C
C WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
  WRITE(6,52) I=1,1
  IF(RATIC(I).GE.RATIC(J)) GC TC 30
  BUFTM=TIME(I)
  BUFHT=WEIT(I)
  BUFTT=RATIC(I)
  I2=J+1
  J1=1
  20 IRANK(J1)=IRANK(J1-1)
  TIME(J1)=TIME(J1-1)
  WEIT(J1)=WEIT(J1-1)
  RATIC(J1)=RATIC(J1-1)
  J1=J1-1
IF(J1.LT.12) GO TO 25
GO TO 25
25 IRANK(J)=1
TIME(J)=BLF1M
WEIT(J)=BLF1W
RATIC(J)=BLFRT
GO TO 4C
30 CONTINUE
IRANK(I)=I
40 CONTINUE
C. IRANK(I)=K ; I-TH POSITION(RANK) IS JCB K
C1; END: END OF ORIGINAL T(I)/m(I) RANK
C
C2; START: ARRANGING ALL JCBS WITH NEW INDEX
C
WRITE(6,51)
51 FORMAT(2X,'JCB',5X,'TIME',4X,'HEIGHT',2X,'RATIC')
DO 53 I=1,N
WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
52 FORMAT(3X,13,5X,F5.1,3X,F5.1,3X,F8.4)
53 CONTINUE
C ; END
DO 57 J=2,N
57 IPRM(I,J)=1
NN=N-1
DO 60 I=2,NN
JJ=I+1
DO 60 J=JJ,N
IF(IPRM(I,J).EQ.C) GC TC 80
60 CONTINUE
ISET(1)=1

C.
NMFTS=1
N1=N-1
DO 75 I=2,N1
JJ=I+1
DO 75 J=JJ,N1
IF(IPRM(I,J).EQ.C) GC TC 80
70 CONTINUE
ISET(1)=1
NMFTS=NMFTS+1
75 CONTINUE
80 ISET(I-1)=-(I-1)
DO 90 K=1,N1
JJ=K+1
DO 85 J=JJ,N1
IF(IPRM(K,J).EQ.O) GC TC 87
85 CONTINUE
ISET(K)=-K
GO TO 90
87 ISET(KJ)=K
90 CONTINUE

C.
N2=N-2
N1=N-1
DO 110 I=1,N2
JJ=I+1
DO 100 J=JJ,N1
IF(1PRM(I,J).EQ.C) GO TO 100
KK=J+1
DO 95 K=KK,N
IF(1PRM(I,K).EQ.1.AND.1PRM(J,K).EQ.1) 1PRM(I,K)=0
95 CONTINUE
100 CONTINUE
110 CONTINUE
DO 130 I=2,N
ISLM=1
JJ=I-1
DO 120 J=1,JJ
IF(1PRM(J,I).EQ.0) GO TO 120
ISLM=ISLM+1
120 CONTINUE
IMRMI, ISLM=J
130 CONTINUE

C. INITIAL SCL. FRGM NATURAL ORDER
ICUMB=0
DO 135 I=1,N
KK=55555
DO 134 J=1,ITLNM
KK1=ICTMA(J)
IF(KK1.GE.KK) GO TO 134
KK=KK1
MN=J
134 CONTINUE
ICTMA(MN)=ICTMA(MN)+TIME(I)
ICUMB=ICUMB+ICTMA(MN)*WET(I)
IJ=M(MN,1)
M(MN,1)=IJ+1
135 M(MN,3+IJ)=1
WRITE(6,136) ICUMB
136 FORMAT(/,5X,*INITIAL SCL. FRGM NATURAL ORDER, ICUMB=*,I8)
DO 137 I=1,ITLNM
KK=M(I,1)
KK2=KK+2
137 WRITE(6,280) I,(M(I,J),J=1,KK2)

C. UPDATE SCL. IN ISCL(I)
ISUM=0
DO 138 I=1,ITLNM
ITL=M(I1,1)
IFJ=ITL+2
ISUM=ISLM+1
ISCL(I1,LM)=ITL
M(I1,1)=0
ICTMA(I1)=0
GO 138 J=3, IFJ
ISUM=ISLM+1
ISCL(I1,LM)=M(I1,J)
138 CONTINUE
C.
C. INFORMATION FOR FAT=CM5 ; START
NN=N-1
BOX=WEIT(NN+1)*TIME(NN+1)*(ITLNM-1)/(2*ITLNM)
WT=WEIT(NN+1)
IFRT=WT*TIME(NN+1)/ITLM
JLBSh(NN)=WT
JLBCB(NN)=IFRT*BCX
142 NN=NN-1
BOX=BCX*WEIT(NN+1)*TIME(NN+1)*(ITLNM-1)/(2*ITLNM)
WT=WEIT(NN+1)+WT
IFRT=WT*TIME(NN+1)/ITLM+IFRT
JLBSh(NN)=WEIT(NN+1)*JLBSh(NN+1)
JLBCB(NN)=IFRT*BCX
IF(NN+GE=2) GC TO 142
C. INFORMATION FOR FAT=CM5 ; END
C.
C. TO GET PARTIAL OPTIMAL SCL BEFORE BRANCHING: FOR S1
ICCST=C
GO 150 I=1, ITLM
M(I,1)=1
M(I,3)=1
ISTJB(I)=0
ICTMA(I)=TIME(I)
ICCST=ICCST+TIME(I)*WEIT(I)
IF(SET(I).LT.0) GC TO 160
150 CONTINUE
160 ISJID=I+1
C.
C. **** BRANCHING ****
C.
ISET(N)=-N
NN=N-1
DO 165 J=ISJID, N
165 MACUJ(J)=0
INTJ8=ISJID
NODE=0
C. INITIALIZE THE CLOCK (TO ZERO)
CALL SCLK1
170 CONTINUE
IBACK=C
DQ 230 I = INTJB,N
NODE=ACCE+1
JOB=ISE(I)
IF(JCB.LT.0) CALL FATHM1(JCB,ICTPA,NUM,S18G)
NUM=MACUJ(JCB)+1
IF(NUM.GT.JCB) CALL ERRCR(120,8100)
180 MACUJ(JCB)=NUM
NJB=M(NUM,1)
M(NUM,NJB+3)=JCB
M(NUM,1)=NJB+1
ISTJBE(JCB)=ICTMA(MNUM)
ICTM1(MNUM)=ICTMA(MNUM)+TIME(JCE)
C.
ICT=ICTPA(MNUM)*HEIT(JCB)
ICCST=ICT+ICCST
RATIC(JCB)=ICT
C.
C.
C. FATHMING PLACE
190 CONTINUE
CALL FATHM2(JCB,IPRM,ISTJBE,82CC)
IF(JCB.EQ.N) GC TC 225
C.
CALL FATHM5(JCB,ICCST,ICUMB,ICTMA,JB5K,JLBGB,82CC)
C.
MCRC BRANCHING
IF(IBACK.EQ.0) GC TC 230
INTJB=JCB+1
GO TC 170
C.
IF THE NCCE IS TO BE FATHMNG,
C. DECIDE WHETHER BACKTRACK OR MCVE-TX-NEXT-MACHINE TAKES PLACE
C.
200 CONTINUE
IF(ISTJBE(JCB).LT.0) GC TC 210
NUM=MACUJ(JCE)
IF(JCE.LE.ITALM.AND.JOB.LT.MNUM) GC TC 220
IF(JCE.LE.ITALM.AND.JOB.GT.MNUM) GC TC 203
IF(JCB.GT.ITALM.AND.MNUM.GT.ITALM) GC TC 22C
IF(JCB.GT.ITALM.AND.MNUM.GT.ITALM) CALL ERRCR(2CC,810CC)
C.
C. MOVE THE JCB TO NEXT MACHINE
C. NEWM: NEW MACHINE 
203 CONTINUE
IDENT=ISTJBE(JCB)
NEWM=MNUM+1
MACUJ(JCB)=NEWM
M(MNUM,1)=M(MNUM,1)-1
ICTMA(MNUM)=ISTJBE(JCB)
NJB=M(NEWM,1)
M(NEWM,11)=NJB+1
M(NEWM,AJB+3)=JCB
ISTJE(JCB)=ICTMA(NEWM)
ICTMA(NEWM)=ICTMA(NEWM)+TIME(JCB)

C.
ICCST=ICCST-RATIC(JCB)
ICT=ICTMA(NEWM)*WEIT(JCB)
RATIC(JCB)=ICT
ICCST=ICCST+ICT

C.
IF(IDENT.EQ.ISTJE(JCB)) GO TC 200
C.
NODE=NGCE+1
C.
GO TC 190
C.
BACKTRACK
C.
UPDATE MACHINE
210 CONTINUE
JOB=-1SET(JCB)
MNLM=MACUJ(JCB)
220 CONTINUE
M(MNUM,1)=M(MNUM,1)-1
ICTMA(MNUM)=ISTJE(JCB)
MACUJ(JCB)=0
C.
ICCST=ICCST-RATIC(JCB)
C.
IBACK=1
C.
WRITE(6,223) JCB,MNLM
223 FORMAT(// BACKTRACK, CURRENT JCB# & CURRENT MACH.#,15,14)
JCB=JCB-1
IF(JCB.LT.ISJ1D) GO TO 240
GO TC 200
C.
CALCULATE CCST FOR CURRENT SCL.
225 NCGST=0
CO 226 IL=1,N
226 NCGST=NCGST+(ISTJE(IL)+TIME(IL))*WEIT(IL)
IF(NCGST.GE.ICUMB) GO TC 200
ICUMB=NCGST
C.
UPDATE SCLLLIGN
ISLM=0
CO 227 IL=1,ITLM
ITL=M(IL,1)
IFJ=ITL+2
ISUM=ISLM+1
ISOL(ISLM)=ITL
CO 227 J=3,IFJ
ISUM=ISLM+1
ISOL(ISLM)=M(IL,J)
227 CONTINUE
C. FATHMING

GO TO 260

230 CONTINUE

240 CONTINUE

ISUM=0
DO 250 I=1,ITLNM
ISLM=ISLM+1
ITL=ISCL(I,SUM)
INDX=2
IIJ=1+ISUM
IFL=ITL+ISUM
M(I,1)=ITL
DO 250 J=IIJ,IFL
ISLM=ISLM+1
JOB=ISCL(I,J)
INCX=INCX+1
M(I,INDX)=JOB
250 CONTINUE

C. REACT THE CURRENT TIME IN SEC.

ACTM=RCLCK11(I,J)
WRITE(6,260) NCDE,ACTM,ICLME

260 FORMAT(/,2X,'**** TITLE # CF NCDE#I7,13//,2X, 
* *** ACTUAL CPL TIME AFTER INITIALIZATION',F12.5,/,2X, 
* *** OPTIMAL CSSY IS',I10)

DO 270 I=1,ITLNM
KK=M(I,1)
KK2=KK+2
WRITE(6,280) I,(M(I,J),J=1,KK2)
270 CONTINUE

280 FORMAT(/,4X,'M=',I3,6X,5Q14)

1000 CONTINUE

RETURN

END

SUBROUTINE FATHM1(JCB,ICTMA,MCUK,*)
CGMCH A,ITLNM,ILAC1,ILAC2
INTEGER*2 ICTMA(ITLNM)
JCB=JCB
MINT=999999999
DO 10 I=1,ITLNM
IF(MINT.LE.ICTMA(I)) GC TG 10
MINT=ICTMA(I)
MNUM=I
10 CONTINUE

RETURN

END

SUBROUTINE FATHM2(JCB,IPRM,ISTJB,*)
CGMCH A,ITLNM,ILAC1,ILAC2
INTEGER*2 IPRM(N,N),ISTJB(N)
KK=IPRM(JCB,1)
KK1=KK+1
10 CONTINUE
C. MORE BRANCH OR NEXT FATHOMING CRITERIA(3)
RETURN
END
SUBROUTINE FATHM5(JOB, ICCST, ICUMB, ICTMA, JLBSH, JLBCH, *)
COMMON N, ITLNM, ILNCl, IUNC2
DIMENSION JLBCB(N)
INTEGER ICTMA(N), ICTMA(1)
10 CONTINUE
RETURN
END
SUBROUTINE ERROR( NER, *)
WRITE(6,1C)(NER)
RETURN
END
/*
//
K.2 LIST OF PROGRAM 2

COMMON/CA/ \# ITLN, IUNC1, IUNC2, IUNC3, IUNC4
C. N: \# CF JCE
C. ITLN: TCTLE \# CF MACHINES
CALL SUBITIME, WEIT, RATIC, JLCBP, IRANK, IPRM, ISET, M, ISCL,
  MACUJ, JLSH, ISTJB, ICTMA, IRCST, IRJBL, IRJBL, ISIC, JBINL, JBINL,
  ICCST, ILCB, IPT, ISUPC, ISUP1, NINT
STOP
END
SUBROUTINE SUBITIME, WEIT, RATIC, JLCBP, IRANK, IPRM, ISET, M,
  ISCL, MACUJ, JLSH, ISTJB, ICTMA, IRCST, IRJBL, IRJBL, ISIC, JBINL,
  JBINU, ICCST, ILCB, IPT, ISUPC, ISUP1, NINT)
CCMN/CA/ \# ITLN, IUNC1, IUNC2, IUNC3, IUNC4
DIMENSION TIME(N), RATIC(N), JLBCB(N)
  ICCST(N), ILCB(N)
INTEGER*2 IRANK(N), IPRM(N, N), ISET(N), M(4*ITLN, IUNC1)
  ISCL(IUNC2), MACUJ(N), JLSH(N), ISTJB(N), ICTMA(4*ITLN)
  IRCST(IUNC3), IRJBL(IUNC3), IRJBL(IUNC3, ISIC(IUNC3)
  JBINL(N), JBINU(N), IPT(N), ISUPC(N, IUNC4), ISUP1(N, IUNC4)
  NINT(N)
N=10
ITLN=N=4
IUNC1=N+2
IUNC2=N+ITLN
IUNC3=N*(N-1)/2
IUNC4=IUNC3
N1=N
N2=N
N3=N
N4=N
N5=(N+1)/2
N6=(N*N+1)/2
N7=N5
N8=(ITLN*IUNC1+1)/2
N9=(IUNC2+1)/2
N10=N5
N11=N5
N12=N5
N13=(ITLN+1)/2
N13=(N*(N-1)/2+1)/2
N14=N13
N15=N13
N16=N13
N17=N13
N18=(N+1)/2
N19=N18
N20=N
N21=N
N22=N18
N23=(N*IUNC4+1)/2
N24=N23
N25=N18
CALL GETK(TIME,N1,HEIT,N2,RATIC,N3,JBLCB,N4,IRANK,N5,
  + IPRM,N6,ISET,N7,PM,N8,ISCL,N9,MACUJ,N10,JLBSH,N11,
  + ISTJ,B,N12,ICTMA,N13,IRCST,N14,IRJBL,N15,IRJBL,N16,
  + IS10,N17,JEINL,N18,JEINC,N19,ICCST,N20,ILCE,N21,
  + IPR,N22,ISLPO,N23,ISUP1,N24,NMINT,N25)

CALL ACTLTIME,HEIT,RATIC,JBLCB,IRANK,IPRM,ISET,M,
  + ISCL,MACUJ,JLBSH,ISTJ,B,ICTMA,IRCST,IRJBL,IRJBL,ISIC,
  + JBINL,JBINU,ILCST,ILCE,IPR,ISLPC,ISLPC,NMINT)
CALL FREEMTIME,HEIT,RATIC,IRANK,N3,JBLCB,N4,IRANK,N5
  + IPRM,N6,ISET,N7,PM,N8,ISCL,N9,MACUJ,N10,JLBSH,N11,
  + ISTJ,B,N12,ICTMA,N13,IRCST,N14,IRJBL,N15,IRJBL,N16,
  + IS10,N17,JEINL,N18,JEINC,N19,ICCST,N20,ILCE,N21,
  + IPR,N22,ISLPO,N23,ISUP1,N24,NMINT,N25)
RETURN
END
SUBROUTINE ACTLTIME,HEIT,RATIC,JBLCB,IRANK,IPRM,ISET,
  + M,ISCL,MACUJ,JLBSH,ISTJ,B,ICTMA,IRCST,IRJBL,IRJBL,ISIC,JBINL,
  + JBINU,ILCST,ILCE,IPR,ISLPC,ISLPC,NMINT)
CMMCN/CA/ N,ITLNM,IUNC1,IUNC2,IUNC3,IUNC4
CMMCN/CA/ NTMIN,INPMX,INPMX,ITLCI
DIMENSION TIME(N),HEIT(N),RATIC(N),JBLCB(N),
  + IOST(IN),ILCO(IN)
INTEGER IRANK(N),IPRM(N,N),ISET(N),ITLNM,IUNC1
  + ISCL(IUNC2),MACUJ(N),JLBSH(N),ISTJ,B,ICTMA(ITALM)
  + IRCST(IUNC3),IRJBL(IUNC3),IRJBL(IUNC3),IS10(IUNC3),JEINL(N)
  + JBINL(N),JBINU(N),ISLPO(N),ILAC4,NMINT(N)
C.
C. NMFTS: # OF JOBS IN 1-ST SET IN S1, S1
C. ICUMB: CCST FOR A INCLMBENT SCLUTION
K2=999
CO 1 I=1,N
2 K1=K2
  CALL KERAND(K1,K2,F)
  IF(IIZ.EQ.0) GC TC 2
3 K1=K2
  CALL KERAND(K1,K2,F)
IK=F*(IC**2)
IF(IK.EQ.0) GC TC 3
TIME(I)=I2
IK=IK-1
C. TO CALCULATE "NTMIN"
C. NTMIN: MIN. TOTAL # CF INTERACTIONS INITIALLY
C. NMMXJ: # MACHINES FOR MAX. JCBS IN NTMIN
C. INMXJ: # JCBS ALLOCATED MAXIMALLY IN CASE OF NMMXJ=0
C. ITLCI: TOTAL # CF INTERACTIONS BEING USED CURRENTLY
   CALL MINIT(IN,ITLNP,INMXJ,NMMXJ,NTMIN)
C.
   ITLCI=NTMIN
C.
C1: START=RANKING THE JCBS WITH RESPECT TO NONDECREASING ORDER OF I
C
RATIC(I)=TIME(I)/HEIT(I)
I=1
IRANK(I)=1
DO 40 I=2,N
RATIC(I)=TIME(I)/HEIT(I)
I1=I-1
DO 30 J=1,II
IF(RATIC(I)*GE.RATIC(J)) GC TC 30
BUFTM=TIME(I)
BUFWT=HEIT(I)
BUFR=RAIC(I)
I2=J+1
J1=I
20 IRANK(J1)=IRANK(J1-1)
TIME(J1)=TIME(J1-1)
HEIT(J1)=HEIT(J1-1)
RATIC(J1)=RATIC(J1-1)
J1=J1-1
IF(J1.LT.I2) GO TC 25
GC TC 20
25 IRANK(J)=I
TIME(J)=BUFTM
HEIT(J)=BUFWT
RATIC(J)=BUFR
GO TC 40
30 CONTINUE
IRANK(I)=1
40 CONTINUE
42 FORMAT(2X,51I4)
C. IRANK(I)=K ; I-TH POSITION(IRANK) IS JCB K
C1; ENC=END CF ORIGINAL I(I)/H(I) RANK
C
C2: START=ARRANGING ALL JCBS WITH NEW INDEX
C
WRITE(*,51)
C2; END
C3 START; PRECEDENCE RELATIONSHIP AMONG JCBS
DO 57 J=2,N
57 IPRM(I,J)=1
NN=N-1
DO 60 I=2,NN
JJ=I+1
DO 60 J=JJ,N
IPRM(I,J)=0
IF(TIME(I).LE.TIME(J).AND.HEIGHT(I).GE.HEIGHT(J)) IPRM(I,J)=1
60 CONTINUE
ISET(I)=1
DO 67 I=1,N
67 CONTINUE
68 FORMAT(2X,14,5X,5CI4)
C.
NMFTS=1
N1=N-1
DO 75 I=2,N1
JJ=I+1
DO 75 J=JJ,N1
IF(IPRM(I,J).EQ.C) GC TG 80
70 CONTINUE
ISET(I)=1
NMFTS=NMFTS+1
75 CONTINUE
80 ISET(I-1)=-(I-1)
DO 90 K=1,N1
JJ=K+1
DO 85 J=JJ,N1
IF(IPRM(K,J).EQ.C) GC TG 87
85 CONTINUE
ISET(K)=-K
GC TG 5C
87 ISET(K)=K
90 CONTINUE
C.
WRITE(6,92) NMFTS,(ISET(I),I=1,N)
92 FORMAT(2X,NMFTS,ISET*,13,2X,5CI4)
N2=N-2
N1=N-1
DO 110 I=1,N2
JJ=I+1
DO 100 J=JJ,N1
IF(IPRM(I,J).EQ.C) GC TG 100
KK=J+1
GO to 95
IF(IPRM(I,K).EQ.1.AND.IPRM(J,K).EQ.1) IPRM(I,K)=C
95 CONTINUE
CONTINUE
DO 130 I=2,N
ISLM=I
JJ=I-1
DO 120 J=1, JJ
IF(IPRM(J,I).EQ.C) GC TO 120
ISLM=ISLM+1
IPRM(I,ISLM)=J
120 CONTINUE
IPRM(I,1)=ISLM-1
CONTINUE
WRITE(6,65) (I,I=1,N)
CONTINUE
WRITE(6,68) I,(IPRM(I,J),J=1,N)
CONTINUE
C3; END
C4. START; INITIAL SCL. FROM NATURAL CRDER
C4. START; AND CALCULATE *IDISM*=SLM CF IT(I)*w(I) FOR ALL I
ICUMB=0
IDISM=C
DO 135 I=1,N
RATIC(I)=HEIT(I)*TIME(I)
IDISM=IDISM+RATIC(I)
MKMT(I)=C
KK=99999
DO 134 J=1,ITLNK
KK1=ICMA(J)
IF(KK1.GE.KK) GC TO 134
KK=KK1
MN=J
134 CONTINUE
ICMA(MA)=ICMA(MA)+TIME(I)
ICUMB=ICUMB+ICMA(MA)*HEIT(I)
IJ=M(MN,1)
M(MN,1)=IJ+1
135 M(MN,3+IJ)=I
WRITE(6,136) ICUMB
136 FORMAT(/,5X,*INITIAL SCL. FROM NATURAL CRDER, ICUMB=' ,18)
DO 137 I=1,ITLNK
KK=M(I,1)
KK2=KK+2
137 WRITE(6,280) I,(M(I,J),J=1,KK2)
C. UPDATE SCL. IN ISGL(I) FROM NATURAL CRDER
ISULM=C
DO 138 I=1,ITLNK
ITL=M(I1,1)
IFJ=ITL+2
ISUM=ISLM+1
ISCL(ISLM)=ITL
M(I1,1)=0
ICTMA(I1)=0
DO 138 J=3,IFJ
ISUM=ISLM+1
ISCL(ISLM)=M(I1,J)
138 CONTINUE
C4 END
C5. START; INFORMATION FCR FATHCM5
NN=N-1
BOX=WEIT(NN+1)*TIME(NN+1)\((ITLN\text{-}1)/(2*ITLM)\)
WT=WEIT(NN+1)
IFRT=WT*TIME(NN+1)/ITLM
JLBSH(NN)=WT
JLBCB(NN)=IFRT+BCX
142 NN=NN-1
BOX=BCX+WEIT(NN+1)*TIME(NN+1)\((ITLN\text{-}1)/(2*ITLM)\)
WT=WEIT(NN+1)+WT
IFRT=WT*TIME(NN+1)/ITLM+IFRT
JLBSH(NN)=WEIT(NN+1)+JLBSH(NN+1)
JLBCB(NN)=IFRT+BCX
IF(NN.GE.2) GC TC 142
WRITE(6,143) (JLBSH(I1),I1=1,N)
WRITE(6,143) (JLBCB(I1),I1=1,N)
143 FORMAT(2X,'JLBSH,JLBCB',52I7)
C5. END; INFORMATION FCR FATHCM5
C6 START; RANKING INTERACTION CCSI WITH corresponding JCBS
K=0
IRJBL(1)=1
IRJBU(1)=2
N1=N-1
GO 148 J=1,N1
N2=J+1
GO 147 I=N2,N
K=K+1
IRCST(K)=TIME(J)*WEIT(I)
IF(K.EQ.1) GC TC 147
N3=K-1
GO 146 II=1,N3
IF(IRCST(K).GE.IRCST(II)) GC TC 146
IRC=IRCST(K)
N4=II+1
12=K
144 IRCST(12)=IRCST(12-1)
IRJBL(12)=IRJBL(12-1)
IRJBU(12)=IRJBU(12-1)
12=12-1
IF(12.LT.N4) GO TO 145
GO TO 144
145 IRCST(I1)=IRC
IRJBL(I1)=J
IRJBU(I1)=1
GO TO 147
146 CONTINUE
IRJBL(K)=J
IRJBU(K)=1
147 CONTINUE
148 CONTINUE
C6 END; RANKING INTERACTION COST
C.
C6.5 START; INITIALIZE IS10(I)=-1 FOR ALL I AND LBF3
C6.5 START; ANC JB1NL(I) & JB1NU(I) FOR ALL I
KTCTL=K
DO 149 I=1,KTCTL
IS10(I)=-1
149 CONTINUE
IJKCT=C
DO 151 I=1,N
IJKCT=IJKCT+TIME(I)*WEIT(I)
150 FCRMAT(1,3X,13,3X,110,2X,15,2X,15)
LBF3=ICISM
DO 152 I=1,NTMIN
152 LBF3=LBF3+IRCST(I)
C.
DO 153 I=1,N
JB1NL(I)=0
153 JB1NU(I)=0
DO 154 I=1,KTCTL
IF(JB1NL(IRJBL(I)).EQ.0) JB1NL(IRJBL(I))=1
IF(JB1NL(IRJBU(I)).EQ.0) JB1NL(IRJBU(I))=1
JB1NU(IRJBL(I))=1
154 JB1NU(IRJBU(I))=1
C6.5 END
C.
C7. START; TC GET PARTIAL OPTIMAL SCL. BEFORE BRANCHING: FOR SI
ICCSS=C
DO 156 I=1,ITLN
M(I,1)=1
M(I,3)=1
ISTJBL(I)=0
ICTMA(I)=TIME(I)
ICCSS=ICCSS+TIME(I)*WEIT(I)
IPT(I)=NTMIN
IF(ISET(I).LT.0) GC TO 160
158 CONTINUE
160 ISJIC=I+1
C7. END
C. **** BRANCHING ****

1. ISET(N) = -N
2. \( M_N = N - 1 \)
3. DO 165 J = ISJID, N
4. MACUJ(J) = 0
5. INTJB = ISJID
6. NCDE = 0

C. INITIALIZE THE CLOCK (IC ZERG)

        CALL SCLCK1

170 CONTINUE
  IBACK = 0
  DO 230 I = INTJE, N
    NCDE = NCDE + 1
    JCB = ISET(I)
    IF (JCB .LT. 0) CALL FATHM1(JCB, MNUM, ICTMA, &180)
    MNUM = MACUJ(JCB) + 1
    IF (MNUM .LT. JCB) CALL ERRCR(20, &100G)

180 MACUJ(JCB) = MNUM
    NJB = M(MNUM, 1)
    M(MNUM, NJB + 3) = JCB
    M(MNUM, 1) = NJB + 1
    ISTJB(JCB) = ICTMA(MNUM)
    ICTMA(MNUM) = ICTMA(MNUM) + TIME(JCB)

C.

    ICT = ICTMA(MNUM) * WEIT(JCB)
    ICCST = ICT + ICCST
    RATIC(JCB) = ICT

C.

C. FATHCMING PLACE

190 CONTINUE

        CALL FATHM2(JCB, IPRM, ISTJB, &120)
        IF (JCB .EQ. N) GC TC 225

C.

        CALL FATHM5(JCB, ICUMB, ICCST, ICTMA, JLBSh, JLBCB, &120)
        CALL FATHM3(JCB, MNUM, ICLMB, M, RATIC, IRCST, IRJBL, &140
                     + IRJBU, IS1O, JBINL, JBINU, NMINT, KTCTL, IJKCT, &140
                     + CO)

C. MORE BRANCHING

    IF (IBACK .EQ. 0) GC TC 230
    INTJB = JCB + 1
    GO TO 170

C. IF THE NCCE IS TO BE FATHMEC,

C. DECIDE WHETHER BACKTRACK CR MCVE-TC-NEXT-MACHINE TAKES PLACE

200 CONTINUE

    IF (ISET(JCB) .LT. 0) GC TC 210
    MNUM = MACUJ(JCB)
    IF (JOB .LE. ITLAM .AND. JOB .EQ. MNUM) GC TC 220
IF(JOB LE ITLNM AND JOB GT MNUM) GC TO 203
IF(JOB GT ITLNM AND MNUM EQ ITLNM) GC TO 220
IF(JOB GT ITLNM AND MNUM GT ITLNM) CALL ERRCR(220,1360)

C. MOVE THE JCB TO NEXT MACHINE
C. NEWM= NEW MACHINE #

203 CONTINUE
IDENT=ISTJE(JCB)
NEWM=MNUM+1
MACUJ(JCB)=NEWM
M(MNUM,1)=M(MNUM,1)-1
ICTMA(MNUM)=ISTJE(JCB)
JOB=M(NEWM,1)
M(NEWM,1)=NEWM
M(NEWM,NEWM+3)=JCB
ISTJE(JCB)=ICTMA(NEWM)
ICTMA(NEWM)=ICTMA(NEWM)*TIME(JCB)

C. ICCST=ICCST-RATIC(JCB)
ICT=ICTMA(NEWM)*HEIT(JCB)
RATIC(JCB)=ICT
ICCST=ICCST+ICT

C. IF(IDENT.EQ.ISTJE(JCB)) GC TO 200
C. NCCE=NCCE+1
C. MNUM=NEWM
GO TO 150
C. BACKTRACK
C. UPDATE MACHINE

210 CONTINUE
JOB=-ISET(JOB)
MNUM=MACUJ(JCB)

220 CONTINUE
M(MNUM,1)=M(MNUM,1)-1
ICTMA(MNUM)=ISTJE(JCB)
MACUJ(JCB)=0

C. ICCST=ICCST-RATIC(JCB)
C. IBACK=1
JOB=JCB-1
IF(JCB LT ISJID) GC TO 240
GO TO 200

C. CALCULATE COST FOR CURRENT SCL.
225 NCCE=0
DO 226 11=1,II
226 NCCE=NCCE+(ISTJE(11)*TIME(11))*HEIT(11)
IF(NCCST.GE.ICUMB) GC TC 200
ICUMB=NCCST
C. UPDATE SOLUTION
ISLM=0
DO 227 II=1,ITLN
ITL=M(II,1)
IFJ=ITL+2
ISUM=ISLM+1
ISCL(ISUM)=ITL
DO 227 J=3,IFJ
ISUM=ISLM+1
ISOL(ISUM)=M(II,J)
227 CONTINUE
C. FATHOMING
GO TO 260
230 CONTINUE
240 CONTINUE
ISUM=0
DO 250 I=1,ITLN
ISLM=ISLM+1
ITL=ISCL(ISUM)
INDX=2
IIIL=I+ISUM
IFL=ITL+ISUM
M(I,1)=ITL
DO 250 J=IIIL,IFL
ISUM=ISUM+1
JOB=ISCL(J)
INCX=INCX+1
M(I,INDX)=JOB
250 CONTINUE
C. READ THE CURRENT TIME IN SEC.
ACTM=RCCLKK(1,1)
WRITE(6,260) NGDE,ACTM,ICLMB
260 FORMAT(/,2X,4 **** TITLE # CF ACCE*,17,/,6X, *
**** ACTUAL CPU TIME AFTER INITIALIZATION*,F12-5,/,6X, *
**** OPTIMAL CCST IS*,I1C)
DO 27C I=1,ITLN
KK=M(I,1)
KK2=KK+2
WRITE(6,280) I,(M(I,J),J=1,KK2)
27C CONTINUE
28C FORMAT(4X,4 M=4,13,6X.50I4)
1000 CONTINUE
RETURN
END
SUBROUTINE FATHM1(JCB,MNUM,ICTMA,*)
COMMON/CA/ N,ITLN,M,ILNC1,ILNC2,ILNC3,ILNC4
INTEGER*2 ICTMA(ITLN)
JCB=-JCE
MINT=5559999
GO TO 10, ITLN M
IF(MINT LE ICTMA(I)) GC TC 10
MINT=ICTMA(I)
MNUN=1
10 CONTINUE
RETURN
END
SUBROUTINE FATHM2(JCB,IPRM,ISTJE,*)
COMMON/CA/ N,ITLN M,ILNC1,ILNC2,ILNC3,ILNC4
INTEGER*2 IPRM(N,N),ISTJBI(N)
KK=IPRM(JCB,1)
KK1=KK+1
GO TO 1=2,KK1
JCBPR=IPRM(JCB,1)
IF(ISTJB(JOB).LT.ISTJBI(JCBPR)) RETURN
10 CONTINUE
C. MGRE BRANCH OR NEXT FATHCMING CRITERIA(3)
RETURN
END
SUBROUTINE FATHM5(JCB,ICUPB,ICCST,ICTMA,JAC3M,JLBCE,*)
COMMON/CA/ N,ITLN M,ILNC1,ILNC2,ILNC3,ILNC4
DIMENSION JLBCE(N)
INTEGER*2 JLBCE(N),ICTMA(ITLM)
KK=ICTMA(I)
LO 10 I=2,ITLM
KK1=ICTMA(I)
10 KK=MIN(KK,KK1)
LBF5=ICCST+KK*JLBCE(JOB)+JLBCE(JCB)
IF(LBF5.GE.ICLMB) RETURN
RETURN
END
SUBROUTINE FATHM3(JCB,INL,M,ICLMB,N,RATIC,IRCST,IRJBL,+
IRJBL,IS10,IBINL,JBINL,NNMINT,KTCTL,IKKCT,*)
COMMON/CA/ N,ITLN M,ILNC1,ILNC2,ILNC3,ILNC4
COMMON/CB/ NMINA,INPXJ,NMXJ,ITLNC
DIMENSION RATIC(N)
INTEGER*2 IRCST(ILNC3),IRJBL(ILNC3),IRJBL(ILNC3),IS10(ILNC3),+
+ JBINL(N),JBINL(N),
+ NITLN,ILNC1,NNMINT(N)
C. START; INFORMATION FOR THE POINTER WITH UPDATED # INTERACTIONS
ICF=0
MJCB=N
MMAC=ITLN M
ICCN T=C
INTRC=0
IF(NMNXJ.NE.C) GC TC 10
KJCB=INPXJ
DO 2 I=1,ITLN M
KK=M4(I,1)
IF(KK.LE.KJGB) GC TC 2
INTRC=INTRC+KK*(KK-1)/2
MJCB=MJCB-KK
MMAC=MMAC-1
ICH=1
2 CONTINUE
IF(ICH.EQ.0) GC TC 4
IF(MJCB.LE.MMAC) GC TC 5
CALL MINIT(MJCB,MMAC,K1,K2,K3)
INTRC=INTRC+K3
GO TC 5
4 INTRC=NTMN
5 GO TC 15
10 KJCB=INPXJ+1
KJCB1=KJCB-1
DO 13 I=1,ITALM
KK=M(I,1)
IF(KK.LE.KJCB1) GC TC 13
IF(KK.GT.KJCB) GC TC 12
ICGNT=ICGNT+1
GO TC 13
12 INTRC=INTRC+KK*(KK-1)/2
MJCB=MJCB-KK
MMAC=MMAC-1
ICH=1
13 CONTINUE
IF(ICH.EQ.0) GC TC 14
IF(MJCB.LE.MMAC) GC TC 15
CALL MINIT(MJCB,MMAC,K1,K2,K3)
INTRC=INTRC+K3
GO TC 15
14 IF(ICH.EQ.0.AND.ICGNT.LE.MMXXJ) INTRC=NTMN
IF(ICH.EQ.0.AND.ICGNT.GT.MMXXJ) INTRC=NTMN+ICGNT-MMXXJ
15 CONTINUE
DO 20 I=1,KTCTL
20 INS10(I)=-1
C. END: INFORMATION FOR THE POINTER
INS0=1
INS1=1
IIUP=0
ICT=0
KK=M(INXLM,1)
DO 60 I=1,ITALM
NJCB=M(I,1)
IF(NJCB.EQ.0) GC TC 60
IF(I.EQ.MMUM.AND.KK.EQ.1) GC TC 60
IND=NJCB+2
IF(I.EQ.MMUM) INC=NJCB+1
DO 60 J=3,INC
JOB1=M(I,J)
IA = JBINL(JOB)
IB = JBINL(JCB1)
KK1 = MAXC11A, IE)
IA = JBINL(JOB)
IB = JBINL(JCB1)
KK2 = MINC11A, IE)
DO 30 ILO = KK1, KK2
IF(JOB1, EQ, IRJBL(IIC) AND JCB, EQ, IRJBL(IIC)) GC TC 40
30 CONTINUE
40 IF(I, EQ, MINM) GC TC 50
   IS10(IIC) = 0
   GO TC 60
50 IS10(IIC) = 1
   ITUP = ITLP + 1
   ICT = ICT + IRCST(IIC)
60 CONTINUE
   ITL = INTC - ITLP
   ITL2 = 0
   ITL1 = 1
70 CONTINUE
   IF(IS10(IITL), EQ, -1) GC TC 60
   ITL1 = IITL1 + 1
   GO TC 70
80 ITL2 = ITL2 + 1
   ICT = ICT + IRCST(IITL)
   IF(ITAL2, LT, ITL) GC TC 70
   LBFL3 = ICT + IJKCT
   IF(LBF3, GE, IJUMBE) RETURN 1
   RETURN
END
SUBRTINE MINIT(MA,MN,IC,NNP,IT)
   IQ = NN/NP
   NN = MA - IQ*MN
   NNM = MN - NN
   IT = (IC + 1)*IC + NNM/2 + IC*(IC - 1)*NNM/2
   WRITE(6, I1C) AA, AM, IC, NNP, IT
10 FORMAT(/2X, 'AA', AM, IC, NNP, IT =', 5 I6)
   RETURN
END
SUBRTINE ERRR(INER, *)
   WRITE(6, 10) INER
10 FORMAT('ERRR IS FRCM*, I7')
   RETURN 1
END
/*
*/
K.3 LIST OF PROGRAM 3

COMMCA N, ITLNM, ILNC1, IUNC2
C. N: # OF JCE
C. ITLNM: # OF MACHINES
CALL SUBTIME, HEIT, RATIC, JLCB, IRANK, IPRM, ISET, M, ISLL,
+ MACUJ, JLIBS, ISTJB, ICTMA
STOP
END
CALL GETTIME SUBTIME, HEIT, RATIC, JLCB, IRANK, IPRM, ISET, M,
+ ISCL, MACUJ, JLIBS, ISTJB, ICTMA
CALL ACTL TIME, HEIT, RATIC, JLCB, IRANK, IPRM, ISET, M,
+ ISCL, MACUJ, JLIBS, ISTJB, ICTMA
CALL FREEMITIME, HEIT, RATIC, JLCB, IRANK, IPRM, ISET, M,
+ ISCL, MACUJ, JLIBS, ISTJB, ICTMA
RETURN
END
SUBROUTINE ACTL(TIME,WEIT,RATIC,JLBCE,IRANK,IPRM,ISET,
+ ISCL,MACUJ,JLBSH,ISTJB,ICMP)
COMMON N,ITLNM,ILAC1,ILNC2
DIMENSION TIME(N),WEIT(N),RATIC(N),JLBCE(N)
INTEGER IRANK(N),IPRM(N),ISET(N),ITLNM,ILNC1,
+ ISCL,ILAC2,MACUJ(N),JLBSH(N),ISTJB(N),ICMP

C.
C. NMFTS: # OF JOBS IN 1-ST SET IN S1, S2
C. ICUMB: COST FOR A INCUMBENT SOLUTION
K2=955
GO TO 1
2 K1=K2
CALL KERAND(K1,K2,F)
IZ=F*(1C**2)
IF(IZ.EQ.0) GO TO 2
K1=K2
CALL KERAND(K1,K2,F)
IK=F*(1C**2)
IF(IK.EQ.0) GO TO 3
TIME(I)=IZ
1 WEIT(I)=1
C1: START: RANKING THE JOBS WITH RESPECT TO NONDECREASING CRITERIA CF
C
WRITE(6,51)
RATIC(I)=TIME(I)/WEIT(I)
I=1
WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
IRANK(I)=1
DO 40 I=2,N
RATIC(I)=TIME(I)/WEIT(I)
WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
I1=I-1
GO TO 30 J=1,11
IF(RATIC(J1).GE.RATIC(J)) GO TO 30
BUFRT=TIME(J1)
BUFWT=WEIT(J1)
BUFTM=RATIC(J1)
I2=J+1
J1=I
20 IRANK(J1)=IRANK(J1-1)
TIME(J1)=TIME(J1-1)
WEIT(J1)=WEIT(J1-1)
RATIC(J1)=RATIC(J1-1)
J1=J1-1
IF(J1.LE.I2) GO TO 25
GO TO 2C
30 IRANK(J)=1
35 IRANK(J)=IRANK(J-1)
TIME(J)=TIME(J-1)
WEIT(J)=WEIT(J-1)
RATIC(J)=RATIC(J-1)
J=J-1
IF(J.LE.I2) GO TO 30
GO TO 2C
50 IRANK(J)=I
TIME(J)=BUFMT
WT(J)=BUFWT
RATIC(J)=BUFRT
GO TO 40
30 CONTINUE
IRANK(I)=I
40 CONTINUE
WRITE(6,42) (IRANK(I),I=1,N)
42 FORMAT(2X,100I4)
C. IRANK(I)=K ; I-TH POSITION(RANK) IS JCB K
C1: END; END OF ORIGINAL T(I)/W(I) RANK
C
C2: START; ARRANGING ALL JCBS WITH NEW INDEX
C
WRITE(6,51)
51 FORMAT(2X,'JCB #',5X,'TIME',4X,'WEIGHT',2X,'RATIC')
DO 53 I=1,N
WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
52 FORMAT(3X,5I1,3X,F5.1,3X,F5.1,3X,2E-4)
53 CONTINUE
C; END
DO 57 J=2,N
57 IPRM(1,J)=1
NN=N-1
DO 60 I=2,NN
JJ=I+1
DO 60 J=JJ,N
IPRM(I,J)=0
IF(TIME(I).LE.TIME(J).AND.WEIT(I).GE.WEIT(J)) IPRM(I,J)=1
60 CONTINUE
ISET(I)=1
WRITE(6,65) (I,I=1,N)
65 FORMAT(11X,50I4)
DO 67 I=1,N
WRITE(6,66) I,(IPRM(I,J),J=1,N)
67 CONTINUE
68 FORMAT(2X,14X,5I4)
C
NMFTS=1
N1=N-1
DO 75 I=2,N1
JJ=I+1
DO 75 J=JJ,N1
IF(IPRM(I,J).EQ.0) GO TO 80
70 CONTINUE
ISET(I)=1
NMFTS=NMFTS+1
75 CONTINUE
80 ISET(I-1)=-(I-1)
DO 90 K=1,N1

JJ=K+1
DO 85 J=JJ,N
IF(IPRM(K,J).EQ.0) GC TO 87
85 CONTINUE
ISET(K)=-K
GO TO 5C
87 ISET(K)=K
90 CONTINUE
C.
WRITE(6,92) NMFTS, (ISET(I), I=1,N)
92 FORMAT(2X,NMFTS, ISET*, I=1,13,2X,5014)
A2=N-2
A1=N-1
DO 11C I=1,N2
JJ=I+1
DO 100 J=JJ,N1
IF(IPRM(I,J).EQ.C) GC TO 100
KK=J+1
DO 95 K=KK,N
IF(IPRM(J,K).EQ.1.AND.IPRM(J,K).EQ.C) IPRM(I,K)=C
95 CONTINUE
100 CONTINUE
110 CONTINUE
CG 115 I=2,N
ISUM=1
JJ=I-1
DO 114 J=1,JJ
IF(IPRM(J,I).EQ.C) GC TO 114
ISUM=ISUM+1
IPRM(I,ISUM)=J
114 CONTINUE
IPRM(I,1)=ISUM-1
115 CONTINUE
NN=N-1
DG 119 I=3,NN
JJ=I+1
DO 119 J=JJ,NN
IPRM(I,J)=0
IF(ITIME(I).LE.TIME(J). AND. WEIT(I).GE.WEIT(J)) IPRM(I,J)=1
119 CONTINUE
C.
WRITE(6,65) (I, I=1,N)
DO 120 I=1,N
120 WRITE(6,68) I,(IPRM(I,J), J=1,N)
N1=N-1
N2=N-2
DO 125 I=3,N2
JJ=I+1
DO 121 J=JJ,N1
IF(IPRM(I,J).EQ.1) GC TC 121
J1=J+1
GO 121 K=J1,N
IF(IPRM(J,K).EQ.1) IPRM(I,K)=1
121 CONTINUE
KK=0
JJ=I+1
GO 123 J=JJ,N
IF(IPRM(I,J).EQ.1) GC TC 123
KK=KK+1
IPRM(I,J+KK)=J
123 CONTINUE
IPRM(I,J)=KK
125 CONTINUE
IPRM(N-1,N-1)=0
IF(IPRM(N-1,N).EQ.1) GC TC 126
IPRM(N-1,N-1)=1
IPRM(N-1,N)=N
126 CONTINUE
C.
WRITE(6,65) (1,1=1,N)
GO 132 I=1,N
WRITE(6,66) 1,(IPRM(I,J),J=1,N)
132 CONTINUE
C.
C. INITIAL SCL. FRCM NATURAL CRDER
ICUMB=C
GO 135 I=1,N
ISTJB(I)=0
KK=55555
GO 134 J=1,ITLNM
KK1=ICTMA(J)
IF(KK1.GE.KK) GC TC 134
KK=KK1
MN=J
134 CONTINUE
ICTMA(MN)=ICTMA(MN)+TIME(I)
ICUMB=ICUMB+ICTMA(MN)*WEIT(I)
IJ=M(MN,1)
M(MN,1)=IJ+1
135 M(MN,3+IJ)=I
WRITE(6,136) ICUMB
136 FORMAT(7,5X,'INITIAL SCL. FRCM NATURAL CRDER, ICUMB=',16)
GO 137 I=1,ITLNM
KK=M(1,1)
KK2=KK+2
137 WRITE(6,280) I,(M(I,J),J=1,KK2)
C. UPDATE SCL. IN ISCL(I)
ISUM=0
GO 138 I=1,ITLNM
ITL = M(I1,1)  
IFJ = ITL + 2  
ISLM = ISLM + 1  
ISCL(ISLM) = ITL  
M(I1,1) = 0  
ICTMA(I1) = 0  
DO 138 J = 3, IFJ  
ISLM = ISLM + 1  
ISCL(ISLM) = M(I1, J)  
138 CONTINUE

C. INFORMR ATION FOR FATHCM5; START

AN = N - 1  
BOX = WEIT(AN + 1) * TIME(AN + 1) * (ITLN - 1) / (2 * ITLN)  
WT = WEIT(AN + 1)  
IFRT = hT * TIME(NN + 1) / ITLN  
JLBSW(NN) = WT  
JLBCB(NN) = IFRT + BCX  
142 AN = AN - 1  
BOX = BCX + WEIT(AN + 1) * TIME(NN + 1) * (ITLN - 1) / (2 * ITLN)  
WT = WEIT(AN + 1) + hT  
IFRT = hT * TIME(NN + 1) / ITLN + IFRT  
JLBSW(NN) = WEIT(AN + 1) + JLBSW(NN + 1)  
JLBCB(NN) = IFRT + BCX  
IF(NN .GE. 2) GC TO 142

C. INFORMR ATION FOR FATHCM5; ENC

C. TO GET PARTIAL OPTIMAL SOL. BEFORE BRANCHING; FOR S1

ICCST = C  
DO 150 I = 1, ITLN  
M(I,1) = I  
M(I,3) = 1  
ISTJB(I) = 0  
ICTMA(I) = TIME(I)  
ICCST = ICCST + TIME(I) * WEIT(I)  
IF (ISET(I).LT.0) GC TO 160

150 CONTINUE

160 ISJID = I + 1

C. **** BRANCHING ****

C.  
ISET(N) = -N  
NN = N - 1  
DO 165 J = ISJID, NN  
165 MACU(J) = 0  
INTJB = ISJID  
NODE = 0

C. INITIALIZE THE CLOCK (IC ZERO)

CALL SCLK1

170 CONTINUE
IBACK=C
DO 230 I=INT(JCB,N)
NODE=NCCE+1
JCB=1SET(I)
IF(JCB.LT.0) CALL FATHM1(JCB,ICTMA,MNUM,18C)
MNUM=MACUJ(JCB)+1
IF(MNUM.GT.JCB) CALL ERRCR(20,6100)
180 MACUJ(JCB)=MNL.
NJBJ=M(MNUM,1)
M(MNUM,NJBJ+3)=JCB
M(MNUM,1)=NJBJ+1
ISTJE(JCB)=ICTMA(MNL.
ICTMA(MNUM)=ICTMA(MNL.)*TIME(JCB)

ICT=ICTMA(MNUM)*WEIT(JCB)
ICGST=ICT+ICGST
RATIC(JCB)=ICT

C. FATHOMING PLACE
190 CONTINUE
CALL FATHMJ2(JCB,IPRM,ISTJB,62CG)
IF(JCB.EQ.N) GC TC 225

C. IF(JCB.LT.3) GC TC 192
CALL REMJP(JCB,ISTJB,ICTMA,IPRM,62CG)
192 CONTINUE
CALL FATHM5(JCB,ICGST,ICLMB,ICTMA,JLBCH,61CG)
C. MCRE BRANCHING
IF( IBACK.EQ.C) GC TC 230
INTJB=JCB+1
GO TC 170
C. IF THE NCCE IS TO BE FATHCMEC,
C. DECIDE WHETHER BACKTRACK OR MOVE—TC—NEXT—MACHINE TAKES PLACE
C.
200 CONTINUE
IF(ISET(JCB).LT.0) GC TC 210
MNUM=MACUJ(JCB)
IF(JCB.LE.ITALM.AND.JOB.EQ.MNUM) GC TC 220
IF(JCB.LE.ITALM.AND.JOB.GT.MALM) GC TC 203
IF(JCB.GT.ITALM.AND.MNUM.EQ.ITALM) GC TC 220
IF(JCB.GT.ITALM.AND.MNUM.GT.ITALM) CALL ERRCR(20,61CG)

C.
C. MOVE THE JCB TO NEXT MACHINE
C. NEWM: NEW MACHINE #
203 CONTINUE
IDENT=ISTJB(JCB)
NEWM=MALM+1
MACUJ(JCB)=NEWM
M(MNUM,1)=M(MNUM,1)-1
ICTMA(MNUM)=ISTJB(JCB)
NJB=M(NEWM,1)
M(NEWM)=NJB+1
M(NEWM,NJB+3)=JCB
ISTJB(JCB)=ICTMA(NEWM)
ICTMA(NEWM)=ICTMA(NEWM)+TIME(JCB)

C.

ICCST=ICCST-RATIC(JCB)
ICT=ICTMA(NEWM)+HEIT(JCB)
RATIC(JCB)=ICT
ICCST=ICCST+ICT

C.

IF(IDENT.EQ.ISTJB(JCB)) GC TC 260
C.

NO=NO+1
C.

GO TC 150
C.

BACKTRACK
C. UPDATE MACHINE
210 CONTINUE

JCB=-ISET(JCB)
MNUM=MACUJ(JCB)

220 CONTINUE

M(MNUM,1)=M(MNUM,1)-1
ICTMA(MNUM)=ISTJE(JCB)
MACUJ(JCB)=0
ISTJE(JCB)=0

C.

ICCST=ICCST-RATIC(JCB)
C.

IBACK=1
JOB=JCB-1
IF(JCB.LE.ISJID) GC TG 240
GO TG 260
C. CALCULATE CGST FOR CURRENT SCL.
225 NCOST=0
DO 226 II=1,N
226 NCOST=NCOST+(ISTJB(II)+TIME(II)+HEIT(II))
IF(NCOST.GE.ICUMB) GC TG 200
ICUMB=NCOST
C. UPDATE SCCUITION

ISUM=C
DO 227 IL=1,NLNM
ILN=M(IL,1)
IFJ=ILN+2
ISLM=ISLM+1
ISCL(ISLM)=ILN
DO 227 J=3,IFJ
ISUM=ISUM+1
ISCL(ISLM)=M(I1,J)
227 CONTINUE

C. FATHOMING
GO TO 2CO
230 CONTINUE
240 CONTINUE
ISLM=0
DO 250 I=1,ITLNM
ISLM=ISLM+1
ITL=ISCL(ISUM)
INDX=2
IIL=1+ISUM
IFL=ITL+ISUM
M(I,1)=ITL
DO 250 J=IIL,IFL
ISUM=ISUM+1
JCB=ISCL(J)
INDX=INDX+1
M(I,INDX)=JCB
250 CONTINUE

C. READ THE CURRENT TIME IN SEC.
ACTM=RCLK(1,1.)
WRITE(6,260) AGDE,ACTM,ICLMB
260 FORMAT(/,2X,"***** TCTLE # CF NNCE*,17x/,2X*
* "***** ACTUAL CPL TIME AFTER INITIALIZATION*,13.5,,2X*
* "***** OPTIMAL CCST IS*,110)
DO 270 I=1,ITLNM
KK=M(I,1)
KK2=KK+2
WRITE(6,280) I,(M(I,J),J=1,KK2)
270 CONTINUE
280 FORMAT(4X,"M=",I3,6X,50I4)
1000 CONTINUE ..
RETURN
END

SUBROUTINE FATHM1(JCB,ICTMA,MNUM,*)
COMMON N,ITLNM,IUNC1,IUNC2
INTEGER*2 ICTMA(ITLNM)
JCB=-JCE
MINT=SS9999SS
DO 10 I=1,ITLNM
IF(MINT.EQ.ICTMA(I)) GC TO 10
MINT=ICTMA(I)
MNUM=I
10 CONTINUE
RETURN 1
END

SUBROUTINE FATHM2(JCB,IPRM,ISTJB,*)
COMMON N,ITLNM,IUNC1,IUNC2
INTEGER*2 IPRM(N,N),ISTJB(N)
KK=IPRM(JCB,1)
KK1=KK+1
CG 10 I=2,KK1
JBPRI=IPRM(JCB,1)
IF(ISIJE(JOB).LT.ISIJBE(JCBR)) RETURN 1
10 CONTINUE
C. MORE BRANCH OR NEXT FATHOMING CRITERIA(3)
RETURN
END
SUBROUTINE FATHM5IJCB,ICCST,ICUMB,ICTMA,JLB5R,JLB5B,*
COMMON N,ITLNM,ILNC1,ILNC2
DIMENSION JLB5BIN)
INTEGER*2 JLES,N,JCTMA(N),ITLNM
KK=ICTMA(1)
GO 10 I=2,ITLNM
KK1=ICTMA(I)
10 KK=MIN(KK,KK1)
LBF5=ICCST+KK*JLB5Sh(JOB)+JLB5E(JCB)
IF(LBF5.GE.ICUMB) RETURN 1
RETURN
END
SUBROUTINE REMJBPIJCB,ISTJB,ICTMA,IPRM,*
COMMON N,ITLNM,ILNC1,ILNC2
INTEGER*2 IPRMN,N,ISTJBIN),ICTMA(N)
ISM=55555555555555555
DO 10 I=1,ITLNM
KK=ICTMA(I)
IF(KK.GT.ISM) GO TO 10
IF(KK.EQ.ISM) GO TO 5
ISM=KK
NMIN=1
GO TO 1C
8 NMIN=NMIN+1
10 CONTINUE
KJB=IPRM(JCB,JCB)
DO 30 J=1,KJB
JB=IPRM(JCB,JCB+1)
KK=IPRM(JB,1)
KMAX=C
DO 20 J=1,KK
JBP=IPRM(JB,J+1)
IF(JBP.LE.JCB) GO TO 15
15 KK1=ISTJB(JBP)
KMAX=MAX0(KK1,KMAX)
20 CONTINUE
IF(KMAX.GT.ISM) GO TO 30
IFCW=N-JB-1PRM(JE,JE)+1
IF(NMIN.LE.ICFCW) RETURN
NMIN=NMIN-IFCW
30 CONTINUE
RETURN 1
END
RETURN
END
SUBROUTINE ERROR(AER,*)
WRITE(6,10) AER
10 FORMAT(//,  ERRCR IS FROM*,I7)
RETURN 1
END
/
*/
//
K.4 LIST OF PROGRAM 4

COMMON N, ITLN M, ILNC1, IUNC2
C. N: # CF JCB
C. ITLN M: TGILE # CF MACHINES

CALL SUBL(TIME, WEIT, RATIG, IASJB, IRANK, IPRM, ISET, M, ISCL,
  MACUJ, ILNAS, ISTJB, ICTMA, ICPT)
STOP
END

SUBROUTINE SUBL(TIME, WEIT, RATIG, IASJB, IRANK, IPRM, ISET, M,
  MACUJ, ILNAS, ISTJB, ICTMA, ICPT)

COMMON N, ITLN M, ILNC1, IUNC2
DIMENSION TIME(N), WEIT(N), RATIG(N)
INTEGER*2 IRANK(N), IPRM(N,N), ISET(N), M(ITLN M, ILNC1)
+ , ISCL(IUNC2), MACUJ(N), IASJB(ILNC1), ILNAS(ILNC1), ISTJB(N)
+ , ICTMA(ITLN M), ICPT(N,N)

N = 15
ITLN M = 2
IUNC1 = N + 2
IUNC2 = N + ITLN M
N1 = N
N2 = N
N3 = N
N4 = (IUNC1)/2
N5 = (N + 1)/2
N6 = (N + N + 1)/2
N7 = N5
N8 = (ITLN M* IUNC1 + 1)/2
N9 = (IUNC2 + 1)/2
N10 = N5
N11 = N4
N12 = N5
N13 = (ITLN M + 1)/2
N14 = N6

CALL GETM(TIME, N1, WEIT, N2, RATIG, N3, IASJB, N4, IRANK, N5,
  IPRM, N6, ISET, N7, M, N8, ISCL, N9, MACUJ, N10, ILNAS, N11,
  ISTJB, N12, ICTMA, N13, ICPT, N14)

CALL ACTL(TIME, WEIT, RATIG, IASJB, IRANK, IPRM, ISET, M,
  ISCL, MACUJ, ILNAS, ISTJB, ICTMA, ICPT)

CALL FREEM(TIME, N1, WEIT, N2, RATIG, N3, IASJB, N4, IRANK, N5,
  IPRM, N6, ISET, N7, M, N8, ISCL, N9, MACUJ, N10, ILNAS, N11,
  ISTJB, N12, ICTMA, N13, ICPT, N14)
RETURN
END
SUBROUTINE ACTLTIME, WEiT, RATIC, IASJB, IRANK, IPRM, ISET, + M, ISCL, MACUJ, IUNAS, ISTJB, ICTMA, ICEPT1
COMMON N, ITLMN, ILNCl, IUNC2
DIMENSION TIME(N), HEiT(N), RATIC(N)
INTEGER*2 IRANK(N), IPRM(N,N), ISET(N), M(1TLMN, IUNC1) + , ISCL(IUNC2), MACUJ(N), IASJB(ILNC1), IUNAS(ILNC1), ISTJE(N) + , ICTMA(1TLMN), ICEPT(N,N)

C
C. NMFIS: # OF JCBS IN 1-ST SET IN S1, S1
C. ICLMB: COST FOR A INCUMBENT SOLUTION
K2=999
DO 1 I=1,N
 2 K1=K2
   CALL KERNAND(K1,K2,F)
   1Z=F*1C*2)
   IF(1Z.EC.0) GC TC 2
 3 K1=K2
   CALL KERNAND(K1,K2,F)
   IK=F*1C*2)
   IF(IK.EC.0) GC TC 3
   TIME(I)=1Z
1 WEIT(I)=IK
C1: START: RANKING THE JCBS WITH RESPECT TO NONDECREASING ORDER OF C
C
WRITE(6,51)
RATIC(1)=TIME(1)/WEIT(1)
I=1
WRITE(6,52) I,TIME(I), WEIT(I), RATIC(I)
IRANK(I)=1
DO 40 I=2,N
  4 RATIC(I)=TIME(I)/WEIT(I)
  5 WRITE(6,52) I, TIME(I), WEIT(I), RATIC(I)
  6 I=I-1
  7 DO 30 J=1,II
  8 IF(RATIC(J).GE.RATIC(J)) GC TC 30
  9 BUFTM=TIME(I)
 10 BUFwI=WEIT(I)
 11 BUFRI=RATIC(I)
 12 J2=J+1
 13 J1=1
20 IRANK(J1)=IRANK(J1-1)
   TIME(J1)=TIME(J1-1)
   WEIT(J1)=WEIT(J1-1)
   RATIC(J1)=RATIC(J1-1)
   J1=J1-1

30 CONTINUE
   RETURN
   END
IF(JL.LT.I2) GO TO 25
GO TO 20
25 IRANK(J)=1
TIME(J)=BLF&
WEIT(J)=BUF&
RATIC(J)=BFA
GO TO 4C
30 CONTINUE
IRANK(I)=1
40 CONTINUE
WRITE(6,42) (IRANK(I),I=1,N)
42 FCRMAT(2X,5014)
C. IRANK(I)=K ; I-TH POSITION(RANK) IS JCB K
C1: END; END OF ORIGINAL 1(I)/H(I) RANK
C
C2; START; ARRANGING ALL JCBS WITH NEW INDEX
C
WRITE(6,51)
51 FCRMAT(2X,'JCB #',5X,'TIME',4X,'WEIGHT',2X,'RATIC')
DG 53 I=1,N
WRITE(6,52) I,TIME(I),WEIT(I),RATIC(I)
52 FCRMAT(3X,I3,5X,F5.1,3X,F5.1,3X,F8.4)
53 CONTINUE
C ; END
DO 57 J=2,N
57 IPRM(I,J)=1
MN=N-1
DG 60 I=2,MN
JJ=I+1
DG 66 J=JJ,N
IPRM(I,J)=0
IF(TIME(I).LE.TIME(J).AND.WEIT(I).GE.WEIT(J)) IPRM(I,J)=1
60 CONTINUE
ISET(I)=1
WRITE(6,65) (I,I=1,N)
65 FCRMAT(11X,5CI4)
DG 67 I=1,N
WRITE(6,66) I,(IPRM(I,J),J=1,N)
67 CONTINUE
68 FCRMAT(2X,14,5X,5CI4)
C.
NMFTS=1
N1=N-1
DG 75 I=2,N1
JJ=I+1
DG 70 J=JJ,N
IF(IPRM(I,J).EQ.C) GC TO 80
70 CONTINUE
ISET(I)=1
NMFTS=NMFTS+1
75 CONTINUE
80 ISET(I-1)=-(I-1)
   DO 90 K=1,N1
   JJ=K+1
   DO 85 J=JJ,N
   IF(IPRM(J,K).EQ.0) GC TG 87
85 CONTINUE
   ISET(K)=-K
   GO TO 93
87 ISET(K)=K
90 CONTINUE
C
WRITE(6,92) NMFTS,(ISET(I),I=1,N)
92 FORMAT(2X,'NMFTS,ISET',I3,2X,5I4)
N2=N-2
N1=N-1
   DO 110 I=1,N2
   JJ=I+1
   DO 100 J=JJ,N1
   IF(IPRM(I,J).EQ.0) GC TG 100
   KK=J+1
   DO 95 K=KK,N
   IF(IPRM(I,K).EQ.1.AND.IPRM(J,K).EQ.1) IPRM(I,K)=0
95 CONTINUE
100 CONTINUE
110 CONTINUE
   DO 114 I=2,N
   ISLM=1
   JJ=I-1
   DO 112 J=1,JJ
   IF(IPRM(J,1).EQ.0) GC TG 112
   ISUM=ISLM+1
   IPRM(I,ISUM)=J
112 CONTINUE
   IPRM(I,1)=ISLM-1
114 CONTINUE
   N1=N-1
   DO 118 I=1,N1
   JJ=I+1
   KK=0
   DO 116 J=JJ,N
   IF(IPRM(I,J).EQ.0) GC TG 116
   KK=KK+1
   IPRM(I,I+KK)=11
116 CONTINUE
   IPRM(I,1)=KK
118 CONTINUE
WRITE(6,65) (I,I=1,N)
   DO 132 I=1,N
WRITE(6,68) I,(IPRM(I,J),J=1,N)
CONTINUE

C INITIAL SCL. FROM NATURAL ORDER
ICUMB=0
DO 135 I=1,N
KK=SSSSS
DO 134 J=1,ITLN(M
KK1=ICTMA(J)
IF(KK1.GE.KK) GC IC 134
KK=KK1
MN=J
134 CONTINUE

ICTMA(MA)=ICTMA(MN)+TIME(I)
ICUMB=ICUMB+ICTMA(MA)*HEIT(I)
IJ=M(MN,1)
M(MN,1)=IJ+1
135 M(MN,3+1J)=I
WRITE(6,136) ICUMB

FCRMAT(/,5X,"INITIAL SCL. FROM NATURAL ORDER, ICUMB=",ICUMB)
DO 136 I=1,ITLN(M
KK=M(I,1)
KK2=KK+2
WRITE(6,660) I,(M(I,J),J=1,KK2)
136 FCRMAT(/,5X,"INITIAL SCL. FROM NATURAL ORDER, ICUMB=",ICUMB)
CONTINUE

C UPDATE SCL. IN ISGL(I)
ISUM=0
DO 138 I1=1,ITLN(M
ITL=M(I1,1)
IFJ=ITL+2
ISUM=ISLM+1
ISGL(ISLM)=ITL
M(I1,1)=0
ICTMA(I1)=0
DO 138 J=3,IFJ
ISLM=ISLM+1
ISGL(ISLM)=M(I1,J)
138 CONTINUE

C IUNAS(1)=N
DO 150 I=1,N
150 IUNAS(I+1)=1
C IPT; CURRENT POINTER
C IPJB; CURRENT JCB TO BE BRANCHED FROM
C
NODE=1
IDEPT(1,1)=1
IDEPT(1,2)=1
IDEPT(1,3)=1
IASJB(1)=1
IASJB(2)=1
\begin{verbatim}

IUNAS(1) = N - 1
IUNAS(1+1) = -1
IDPT = 1
MACUJ(1) = 1
M(1,1) = 1
M(1,3) = 1
ISTJB(1) = 0
ICTMA(1) = TIME(1)
RATIC(1) = ICTPA(1) * hEIT(1)
ICCST = 0
ICCST = ICCST + RATIC(1)

C. CREATE NEXT BRANCHING NODE GIVEN ICPI,IASJE(I) AND IUNAS(I)
C.
C. INITIALIZE THE CLOCK (TC ZERC)
CALL SCLK1
205 NGDE = NCDE + 1
IDPT = ICPT + 1
IF(ICPT .EQ. N) GC TC 500
IPJ = ICPF(T(ICPT - 1, 2)
IPJB = IDEPT(ICPT - 1, IPT + 2)
NM1 = ICPF(T(ICPT - 1, 1)
KK = 0
IF(NM1 .EQ. 1) GC TC 230
II = 1
210 IF(II .GE. IPT) GC TC 220
KK = KK + 1
IDEPT(ICPT, KK + 2) = ICPF(T(ICPT - 1, II + 2)
IF(II .GE. NM1) GC TC 230
II = II + 1
GO TO 210
220 IF(II .GE. NM1) GC TC 230
II = II + 1
GO TO 220
230 NM2 = IPRM(IPJB, IPJB)
IF(NM2 .EQ. 0) GO IC 262
DG 260 IK1 = 1, NM2
JB = IPRM(IPJE, IPJE + IK1)
IF(JB .EQ. 2) GC TC 250
NM3 = IPRM(JB, 1)
DG 250 IK2 = 1, NM3
JBPR = IPRM(JB, IK2 + 1)
NM4 = IASJB(1)
DG 240 IK3 = 1, NM4
IF(JBPR .NE. IASJB(1K3 + 1)) GC TC 240
GO TO 250
240 CONTINUE
GO TO 260
250 CONTINUE
KK = KK + 1
\end{verbatim}
IDEPT(ICPT, KK+2) = JB

260 CONTINUE
262 CONTINUE

IDEPT(ICPT, 1) = KK
IDEPT(ICPT, 2) = 1

C.

C. ASSIGN A JCB
JCB = IDEPT(ICPT, 1 + 2)

C. UPDATE MACH. COMPLETION TIME AND STARTING TIME OF JCB
NMJ = IASJB(1)
IASJB(1) = NMJ + 1
IASJB(NMJ + 2) = JCB
NMJ = IUNAS(1)
IUNAS(1) = NMJ - 1
IUNAS(1 + JCB) = -JCB

C. FIND OUT THE SMALLEST COMPLETION TIME OF MACH.
KK = SSSSSSSSS
DO 270 I = 1, ITLN M
KK1 = ICTMA(I)
IF(KK .LE. KK1) GC IC 270
KK = KK1
MNUM = 1
270 CONTINUE

C.

C. ACH THE JCB ON THE Earliest available MACH.
MACU(JCB) = MNUM
NJB = M(MNUM, 1)
M(MNUM, 1 + NB + 3) = JCB
M(MNUM, 1) = NJB + 1
ISTJE(JCB) = ICTMA(MNUM)
ICTMA(MNUM) = ICTMA(MNUM) + TIME(JCB)
RATIO(JCB) = ICTMA(MNUM) * KEIT(JCB)
ICCST = ICCST + RATIC(JCB)

C.

II = IASJE(I) + 1
II = N + 1

C. CHECK RECURSIVITY
CALL FATHMII(JCB, MNUM, N, &3CO)

C.

C. CHECK FATHOMING
CALL FATHMI5(ICCST, ICUMB, ICTMA, IUNAS, TIME, KEIT, &3CO)
GO TC 2C5

C.

C. FATHMED
300 CONTINUE
IF(ICPT .EQ. 1) GC TC 70G
IPT = IDEPT(ICPT, 2)
IND = IDEPT(ICPT, 1)
IF(IPT .EQ. IND) GC TC 400
C.
NEXT JCB AT THE SAME DEPTH
ADDE=ADDE+1
JCBRM=ICEPT(ICPT, IPT+2)
JCBN=ICEPT(ICPT, IPT+3)
MNUM=MACUJ(JCBRM)
MACUJ(JCBN)=MNUM
NJBJ=M(MNUM+1)
M(MNUM,NJBJ+2)=JCBN
ISTJB(JCBN)=ISTJE(JCBRM)
ICTMA(MNUM)=ICTMA(MNUM)-TIME(JCBRM)+TIME(JCBN)
ICCST=ICCST-RATIC(JCBRM)
RATIC(JCBN)=ICTMA(MNUM)*WEIT(JCBN)
ICCST=ICCST+RATIC(JCBN)
C.
UPDATE PGINTER, IASJE, AND IUNAS
IDP=IEPT(ICPT, 2)=IPT+1
C.
M=M+1
CALL FAHMI(JCBN, MNUM, N, 300)
C.
CALL FAHMS(ICCST, ICUMB, ICTMA, IUNAS, TIME, WEIT, &300)
GO TC 205
C.
BACKTRACK
JCBRM=ICEPT(ICPT, IPT+2)
MNUM=MACUJ(JCBRM)
NJBJ=M(MNUM+1)
M(MNUM,1)=NJBJ-1
ICTMA(MNUM)=ISTJE(JCBRM)
ICCST=ICCST-RATIC(JCBRM)
IASJB(1)=IASJB(1)-1
IUNAS(1)=IUNAS(1)+1
IUNAS(1+JCBRM+1)=JCBRM
IDPT=ICPT-1
C.
M=M+1
GO TC 300
500 I=N
510 LJB=IUNAS(I+1)
IF(I=GT.0) GO TC 520
I=I-1
GO TO 510

520 KK=9999999999
DO 530 I=1,ITLNM
   KK1=ICTMA(I)
   IF(KK.LE.KK1) GO TO 530
   KK=KK1
   MNUM=I
530 CONTINUE

C. ADD THE LAST JOB
   ICTMA(MNUM)=ICTMA(MNUM)*TIME(LJB)
   RATIO(LJB)=ICTMA(MNUM)*HEIT(LJB)
   ICCST=ICCSI+RATIC(LJB)
   IF(ICCST.GE.ICUMB) GO TO 550

C. UPDATE THE SCL.
   NJB=M(MNUM,1)
   M(MNUM,NJB+3)=LJB
   M(MNUM,1)=NJ6+1
   ICUMB=ICCSI
   INCODE=NCDE

C.
   ISUM=G
   GO 540 11=1,ITLNM
   ITL=M(I1,1)
   IF=ITL+2
   ISUM=ISLM+1
   ISCL(ISLM)=ITL
   GO 540 J=3,IFJ
   ISUM=ISLM+1
   ISCL(ISLM)=M(I1,J)
540 CONTINUE
   M(MNUM,1)=M(MNUM,1)-1

C.
   ICTMA(MNUM)=ICTMA(MNUM)-TIME(LJB)
   ICCST=ICCST-RATIC(LJB)
   IDPT=IDPT-1
   GO TO 300

600 FORMAT(4X,'*M=',13,6X,5014)
610 FORMAT(12X,'IASJE=',5014)
620 FORMAT(12X,'ILNAS=',5014)
700 CONTINUE

C. READ THE CURRENT TIME IN SEC.
   ACTM=RCLCK1(1.)
   ISUM=0
   DO 705 I=1,ITLNM
      ISUM=ISLM+1
      ITL=ISCL(ISUM)
      INDX=2
      IIL=1+ISUM
705 CONTINUE
IFL=ITL*ISUM
M(I,1)=ITL
DO 705 J=1,1L,IFL
ISUM=ISUM+1
JGB=ISCL(JJ)
INDX=INCA*1
M(I,INDX)=JGB
705 CONTINUE
WRITE(6,710) NODE,ACTM,ICMB
710 FORMAT(//,2X,'**** TCTLE CF NODE',I7,/,2X,'**** ACTUAL CPU TIME A
*FTER INITIALIZATION',F12.5,/,2X,'**** OPTIMAL CCST IS',I1C)
DO 720 I=1,ITLNM
KK=M(I,1)
KK2=KK+2
WRITE(6,610) I,IP(I,J),J=1,KK2
720 CONTINUE
RETURN
DEBUG SLBCHK
END
SUBROUTINE FATHM5(ICCST,ICUMB,ICTMA,IUNAS,TIME,REUT,*)
COMMON N,ITLNM,IUNA1,IUNC2
INTEGER*2 IUNAS(IUNAS(1))
DIMENSION TIME(N),HEIT(N)
INTEGER*2 IUNAS(IUNA1),ICMB(1)
N1=N+1
C1=0.
CK=G.
IP=0
ITIM=0
DO 10 I=2,N1
JB=IUNAS(I)
IF(JB.LT.0) GO TO 10
ITIM=ITIM+TIME(JB)
C1=C1+TIME(JB)*HEIT(JB)
CK=CK+TIME(JB)*HEIT(JB)
IP=IP+HEIT(JB)
10 CONTINUE
KK=S55555555
DO 20 I=1,ITLNM
KK1=ICTMA(I)
20 KK=MIN(KK,KK1)
LB=ICCB+C1/ITLNM*(ITLNM-1)*CK/(2*ITLNM)+KK*IP
IF(LB.GE.ICMB) RETURN 1
RETURN
END
SUBROUTINE FATHM1(JCB,MNLM,*,*)
COMMON N,ITLNM,ILH1,ICMB
INTEGER*2 M(ITLNM,1)
DO 10 I=1,KK

JOB1=M(NUM, I+2)
IF(JOB1.GT.JCB) RETURN 1
10 CONTINUE
RETURN
END
/*
*/