ROOT, James Gordon, 1934-
SCHEDULING WITH DEADLINES AND LOSS FUNCTIONS AND THE OPTIMAL SELECTION OF RESOURCES TO PERFORM TASKS.

The Ohio State University, Ph.D., 1963
Engineering, general

University Microfilms, Inc., Ann Arbor, Michigan
SCHEDULING WITH DEADLINES
AND LOSS FUNCTIONS
AND
THE OPTIMAL SELECTION OF RESOURCES
TO PERFORM TASKS

DISSERTATION
Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By

The Ohio State University
1963

Approved by

[Signature]
Adviser
Department of Industrial Engineering
PREFACE

During my years as a graduate student my main interest has been in the study of management and production problems. Although I am not a mathematician, I have been particularly interested in the use of mathematics in these problem areas. In this dissertation I have chosen two problems and attempted a mathematical treatment. The problems are not completely solved, but I believe significant information and a methodology have been developed to serve as a basis for further research.

Chapters II, III, and IV are concerned with the problem of sequencing $H$ jobs on one machine and $H$ jobs on $K$ machines in parallel. The criterion used in this one stage production is lateness or tardiness. There seems to have been relatively little research on this problem and my aim has been to develop some information on the sequencing problem not only for its own value but to help in establishing a path for further research.

Chapter V is concerned with the selection of resources to perform tasks. The solution technique is based on some ideas developed in a new sequence of courses, offered in the Mathematics Department, on the design of automatic systems. My aim in this chapter has been to show the application of some of these ideas and to carry the solution to the point of current mathematical research.
I would like to express my appreciation to the faculty of the Industrial Engineering Department for providing me with the background and environment so necessary for this undertaking. In particular I would like to thank Professor William T. Morris, my adviser, for his guidance and continuous encouragement not only during the preparation of this dissertation but also during all of my graduate study at The Ohio State University.

I would also like to thank Professor Robert F. Miller for his excellent suggestions which have been incorporated into many areas of this dissertation and also for the many hours he spent with me in getting the theorems into their final form.

Finally, I would like to thank the Battelle Memorial Institute for their financial support which made possible this current year of research and for their clerical help in the preparation of this dissertation.
# 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>General Method For H jobs On One Machine</td>
<td>6</td>
</tr>
<tr>
<td>III</td>
<td>Theorems For H jobs On One Machine</td>
<td>12</td>
</tr>
<tr>
<td>IV</td>
<td>Theorems For H Jobs on K Machines</td>
<td>20</td>
</tr>
<tr>
<td>V</td>
<td>Selection Problem</td>
<td>48</td>
</tr>
<tr>
<td>VI</td>
<td>Conclusions</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>Autobiography</td>
<td>60</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Truth Table</td>
</tr>
<tr>
<td>2</td>
<td>Truth Table</td>
</tr>
</tbody>
</table>
LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Illustration</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sample Problem Schedule</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Schedule</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Schedule</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Schedule</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Schedule</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>Schedule</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>Schedule on Machine j</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>Schedule</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>Schedule</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>Schedule</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>Schedule</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>Transformation of a Schedule</td>
<td>43</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

Many of the problems confronting us today are of the following type. We are given certain tasks to perform and certain resources we might use to accomplish these tasks. The problem is to determine which resources we are going to use and how we are going to use them so that we perform the tasks in the best way. For example, a new company must decide on its plant and equipment and how it may best utilize them in the production of its products.

Because of the complexity of this problem, it is usually partitioned into smaller problems and then an attempt is made to solve these smaller problems. This procedure usually leads to suboptimization, but in many cases this is the best that we know how to do. For example, inventory models and scheduling models are cases of suboptimization since to include all the resources of the company in the models would lead to a situation too complex for analytical treatment.

This dissertation will treat two aspects of the general problem. The first is a special case of sequencing \( H \) jobs on one or \( K \) machines in single stage production using a loss function based on lateness as the criterion. The second is the selection of resources to perform tasks.

The general sequencing problem usually has its setting in the
job-shop. In this problem we assume we are given \( H \) jobs (tasks), \( K \) machines (resources), the job routings, the operation times, \( a_{ki} \) (note \( a_{ki} = a_{k1} \)), for job \( i \) on machine \( k \), the deadlines, \( d_{i} \), for each job (let \( A_{i} \) be the actual completion time), and depending on the exact problem formulation such other information as the transportation times for each job between machines, whether orders may be split, etc. The problem is then to determine how the jobs should be sequenced or scheduled on each machine so that some measure of effectiveness is optimized.

Various criteria have been suggested such as

1. Minimize the total time to complete all jobs.
2. Maximize machine usage.
3. Minimize maximum tardiness, \( \min \max (A_{i} - d_{i}) \).
4. Minimize total tardiness, \( \min \sum (a_{i} - d_{i}) \) for \( A_{i} \geq d_{i} \).
5. Minimize the cost of being late for all jobs. For example, \( \sum c_{i}(A_{i} - d_{i}) \) for \( A_{i} \geq d_{i} \) where \( c_{i} \) is the cost per late time unit. Other loss functions based on lateness have also been suggested.

For this general problem criterion one has been the most widely used. Combinatorial solutions have been limited to cases where each job has at most three machines in its routing. (This is called three stage production.) Programming solutions have been formulated for \( N \) stage production but the computation problem is enormous. Simulation studies have also been made for \( N \) stage production utilizing several of the criteria listed above. Sisson (17), Ackoff (2), and Thompson (20) have copious bibliographies on this work. Sisson (17) reviews all the research to 1957, Ackoff (2) reviews the research to
1961 and Thompson (20) reviews the research from 1957 to 1960. For the special area of simulation Conway (5) has a good discussion and also many references. Of the research mentioned above only that which pertains to single stage production on one or K machines will be discussed in detail.

As Sisson (17) points out the one-machine case for single stage production is not trivial. For example, a computing facility is usually a one-machine problem, and as automation progresses, we may find more one-machine applications. The use of lateness as a criterion seems to be an important one. For example, most products are sold with a delivery date and if this date is missed, it will probably result in an implicit if not an explicit loss.

This criterion has been largely overlooked except for some simulation attempts which are reported by Sisson (17). There may be good reason for this because this criterion seems difficult to apply. However, the job-shop sequencing problem in general seems to have received relatively little attention in relation to its importance.

For the one-machine case Smith (19) has found solutions for minimizing the maximum tardiness and the $\sum (c.i)(A.i)$ where $A.i$ is the completion time and $c.i$ is a weighting constant indicating the value of job i. Also for the one-machine case McNaughton (10) and Schild and Fredman (15) have found a solution when the loss function is linear. Schild and Fredman (16) have also developed some sequencing criteria for a quadratic loss function and a general solution for any loss function which for H jobs requires $H(2^H-1)$ calculations.
For the K-machine case McNaughton (10) shows that the splitting of jobs is not necessary for optimality when the deadlines are all zero and the loss function is linear.

The second problem considered in this dissertation is the selection of the fewest number of resources or the minimum cost resources which will perform a given set of tasks. The technique used in the solution is an application of Boolean algebra. Boolean algebra seems to have found its widest application in the design of electrical circuits. (See (8) and (9) for a discussion of this application.)

There have been several applications of symbolic logic in the Industrial Engineering and Operations Research areas. Ackerman (1) reports on a scheduling application to determine feasible sequences. Ashen (6) gives an example on product engineering and a game problem, and Berkeley (4) cites some examples of determining the consistency and implications of sets of rules. The most recent application reported is by Smidt (18) on a plant location problem. The resource selection problem as treated in Chapter V may also be formulated as a nonlinear programming problem and the reader is referred to the extensive programming literature for these programming techniques.

The literature search included books and periodicals and references contained in Batchelor's Operations Research - An Annotated Bibliography and Operations Research/Management Science abstracts. After the initial search it was found that a few journals contained most of the pertinent articles and these journals were given a more thorough search as indicated: Industrial Engineering, 1952-1963; Quarterly of Applied
In this chapter a general solution technique will be developed for the one-machine case and any loss function.

Let us be given \( H \) jobs, \( i = 1, 2, \ldots, H \), and associated with each job an operation time \( a_i \) and a deadline \( d_i \). Also associated with each job is a loss function \( f_i \) such that if job \( i \) is \( x \) units late the loss is \( f_i(x) \).

It will be assumed in Chapters II, III, and IV that all jobs become available for processing at the same time and that the only jobs that are to be processed on the machine (machines in Chapter IV) are the \( H \) jobs under consideration.

Time zero is the time that work begins on the first job. Once the processing of the jobs begins it will be assumed that each machine operates continuously without idle time on one job at a time until all the jobs assigned to that machine are completed.

The deadlines for the jobs will be measured from time zero. If \( d_i < 0 \), this means that the deadline for job \( i \) has passed before the processing of any of the jobs has begun. If \( d_i \geq 0 \), this means that the deadline for job \( i \) occurs on or after work begins on the first job.

Since the set of operation times, \( (a_1, \ldots, a_H) \), is finite, there exists a largest real number, \( e \), such that \( a_i/e \) is an integer.
for all $i$ although usually not the same integer for all jobs. For job $i$ call this integer $N.i$. For example, if $a.1 = \frac{1}{2}$, $a.2 = 2$, and $a.3 = 3$, then $e = \frac{1}{2}$, $N.1 = 3$, $N.2 = 4$, and $N.3 = 6$. For all $i$ divide job $i$ into $N.i$ parts and label them $a.i.N.i, \ldots, a.i.1$. For all jobs the total number of $a.i.j$'s is $\sum_{i=1}^{H} N.i = N$. In the example above $N = 13$.

Now any sequence of $a.i$'s determines a schedule in the one-machine case and a completion time for each job. Knowing the completion time and the deadline for each job, we can compute the loss for a schedule. In the previous example if $d.1 = 1$, $d.2 = 2$, $d.3 = 3$ and $f.1 = f.2 = f.3 = x$ for $x \geq 0$ and zero otherwise, the sequence $a.1$, $a.2$, $a.3$ yields the loss $f.1(1\frac{1}{2} - 1) + f.2(1\frac{1}{2} + 2 - 2) + f.3(1\frac{1}{2} + 2 + 3 - 3) = \frac{5}{2}$.

This situation is shown in Figure 1.

```
0   1\frac{1}{2}   3\frac{1}{2}   6\frac{1}{2}
\uparrow \quad \uparrow \quad \uparrow
\downarrow d.1 \quad d.2 \quad d.3
```

Figure 1.—Sample problem schedule

For any sequence of $a.i$'s there corresponds exactly one sequence of the $a.i.j$'s formed by substituting $a.i.N.i, \ldots, a.i.1$ in place of $a.i$. In our example the sequence $a.1.3$, $a.1.2$, $a.1.1$, $\ldots$, $a.3.1$ corresponds to the sequence $a.1$, $a.2$, $a.3$. From a sequence of $a.i.j$'s we can compute the actual completion time for job $i$ by simply counting, from the beginning of the sequence, the number of $a.i.j$'s up to and including $a.i.1$ and then multiplying this number by $e$. 
Since every sequence of a₁'s yields a sequence of a₁,j's, it is sufficient to consider sequences of a₁,j's and then try to find that sequence (or that assignment of the a₁,j's to the N time periods of length e) which yields the minimum loss. A dynamic programming approach will be used to solve this problem.

If we define C_k([a₁,j]) to be

\begin{equation}
C_k([a₁,j]) = \text{the minimum loss starting at time period } k \text{ with the set } [a₁,j] \text{ of } N - K + 1 \text{ a₁,j's not assigned},
\end{equation}

then obviously C₁ is the answer to the minimization problem. Since for a given i the a₁,j's are identical except for their labels, it is sufficient to consider only those sequences of a₁,j's in which for each i a₁,j is assigned to a period later than a₁,j+1. With this in mind let us define g(a₁,j) to be

\begin{equation}
g(a₁,j) = f_i(x) \text{ if } j = 1 \text{ and zero otherwise}
\end{equation}

where f_i(x) is the loss when job i is x units late.

To obtain C₁ a recursion relation will be used. Quite clearly

\begin{equation}
C_k([a₁,j]) = \min_{a₁,j} \left[ g(a₁,j) + C_{k+1} ([a₁,j] - a₁,j) \right]
\end{equation}

where the set (a₁,j) is defined to be

\begin{equation}
(a₁,j) = (a₁,h | a₁,h ∈ [a₁,j] \text{ and for } k > h \text{ a₁,k} \notin [a₁,j]).
\end{equation}

The set defined in (2.4) restricts the assignments or sequences to those where for each i if a₁,1 is assigned to period k, then a₁,2 is assigned to an earlier period, a₁,3 to still an earlier period, etc. Since f_i depends only on the period where a₁,1 is assigned, there is no loss in generality in this restriction. It also follows that the splitting of jobs cannot reduce the loss and is not necessary.
To obtain a solution we note that \( C.N + 1 = 0 \) and then recursively compute \( C.N, C.N - 1, \ldots, C.1 \). To compute \( C.k \) requires at most 
\[
\binom{N}{N - K + 1}
\]
subcomputations. This is the number of sets, \([a.i.j]\), of \( N - K + 1 \) a.i.j's that can be formed from \( N \) a.i.j's. We have shown that not all of these sets need to be considered. For example, at period \( N \) we need only consider the sets \([a.i.1]\), \( \ldots, [a.H.1] \). The total number of calculations needed to obtain a solution is at most

\[
\sum_{k=1}^{N} (N - K + 1) = 2^N - 1.
\]

**Sample Problem.**—Suppose we are given three jobs with \( a.1 = 1, a.2 = 2, a.3 = 1, d.1 = 1, d.2 = 2, \) and \( d.3 = 3 \). Let us also assume that \( f.1 = f.2 = f.3 = x^2 \ if \ x \geq 0 \) and zero otherwise. The largest number which yields an integer when divided into one and two is one. Therefore, \( e = 1 \) and job 1 is split into \( a.1.1 \), job 2 into \( a.2.1 \) and \( a.2.2 \) and job 3 into \( a.3.1 \). The calculations for \( C.k \) are

\[
\begin{align*}
C.5 & = 0 \\
C.4(a.1.1) & = (4 - 1)^2 + 0 = 9 \\
C.4(a.2.1) & = (4 - 2)^2 + 0 = 4 \\
C.4(a.3.1) & = (4 - 3)^2 + 0 = 1 \\
C.3(a.1.1, a.2.1) & = \min \left[ (3 - 1)^2 + 4, (3 - 2)^2 + 9 \right] = 8 \\
C.3(a.1.1, a.3.1) & = \min \left[ (3 - 1)^2 + 1, (3 - 2)^2 + 9 \right] = 5 \\
C.3(a.2.1, a.3.1) & = \min \left[ (3 - 2)^2 + 1, (3 - 3)^2 + 4 \right] = 2 \\
C.3(a.2.1, a.2.2) & = 0 + 4 = 4 \\
C.2(a.1.1, a.2.1, a.3.1) & = \min \left[ (2-1)^2 + 2, (2 - 2)^2 + 5, 0 + 8 \right] = 3 \\
C.2(a.1.1, a.2.1, a.2.2) & = \min \left[ (2 - 1)^2 + 4, 0 + 8 \right] = 5 \\
C.2(a.2.1, a.2.2, a.3.1) & = \min \left[ 0 + 2, 0 + 4 \right] = 2 \\
C.1(a.1.1, a.2.1, a.2.2, a.3.1) & = \min \left[ (1 - 1)^2 + 2, 0 + 3, 0 + 5 \right] = 2.
\end{align*}
\]

The minimum loss is two and the optimal sequence is \( a.1.1, a.2.2, a.2.1, a.3.1 \) or job sequence 1, 2, 3. The number of calculations required was eleven.
The main difficulty with this solution method is that the number of computations required for a solution may be very large. In particular the number of calculations depends on the time unit e which the a.i.j's represent. For a given set of jobs and operation times the number of calculations required to obtain a solution is a strictly decreasing function of the time unit e, but depending on the operation times, we are limited as to how large e may be. This follows from the fact that for each $\frac{2i}{e}$ must be an integer. To reduce this dependence on the time unit e and to reduce the number of calculations, the following modification is suggested.¹

For the H jobs let there be H time periods where in each period a job must be assigned. If at time period k jobs i.1, . . . , i.k-1² have been assigned and job i.k is assigned to period k, then the completion time for job i.k is

$$A.i.k = \sum_{j=1}^{k} a.i.j$$

where a.i.j is the total operation time for the job assigned to the jth period. The number of late units for job i.k is

$$x = A.i.k - d.i.k$$

In a manner similar to the previous method define C.k([i]) to be

$$C.k([i]) = \text{the minimum loss starting at time period k with the set } [i] \text{ of } H - k + 1 \text{ jobs not assigned.}$$

¹This modified formulation, discovered independently by the author, is essentially identical to one proposed by Held and Karp (7).

²Note that i.j represents the job number of the job assigned to the j period and is not the jth part of job i.
As before \( C_k \) is a recursive relationship satisfying
\[
(2.8) \quad C_k(\{i\}) = \min_{\{i\}} [f_1(x) + C_{k+1}(\{i\} - i)].
\]
To compute \( C_1 \) we note that \( C_{H+1} = 0 \) and solve for \( C_H, C_{H-1}, \ldots, C_1 \).

For this method the number of subcomputations required at period \( k \) is \( (H-k+1)^H \) and the total number of computations required to obtain a solution is \( \sum_{k=1}^{H} (H-k+1) = 2^{H-1} \). This appears to be an improvement over the \( H(2^{H-1}) \) calculations required in the Schild and Friedman (16) method and when \( H > 3 \), the \( H! \) calculations required for enumeration. A better comparison could be made if efficient computer programs were available for the three methods. However, to obtain these programs would require considerable computer research.

**Sample Problem.**—Using this modified method the computations for the previous sample problem are
\[
\begin{align*}
C.4 & = 0 \\
C.3(1) & = (4 - 1)^2 + 0 = 9 \\
C.3(2) & = (4 - 2)^2 + 0 = 4 \\
C.3(3) & = (4 - 3)^2 + 0 = 1 \\
C.2(1,2) & = \min. [(2 - 1)^2 + 4, (3 - 2)^2 + 9] = 5 \\
C.2(1,3) & = \min. [(3 - 1)^2 + 1, (3 - 3)^2 + 9] = 5 \\
C.2(2,3) & = \min. [(3 - 2)^2 + 1, 0 + 4] = 2 \\
C.1(1,2,3) & = \min. [(1 - 1)^2 + 2, (2 - 2)^2 + 5, 0 + 5] = 2.
\end{align*}
\]
As before the minimum loss is two and the optimal job sequence is 1, 2, 3. This method required seven computations.
CHAPTER III
THEOREMS FOR H JOBS ON ONE MACHINE

Even the improved method of Chapter II may require a large number of calculations and one might ask if a more direct solution method might be found. It is the purpose of this chapter to explore this question. It seems reasonable and it will be assumed in Chapters III and IV that the loss function, \( f_i(x) \), is a strictly increasing function of the number of late time units, \( x \), for \( x > 0 \) and for \( x \leq 0 \) \( f_i(x) = 0 \). This means that as the lateness of a job increases so does the loss and that if a job is completed on or before its deadline, no loss is incurred.

For the purpose of Chapters III and IV the term schedule will now be defined explicitly. A schedule for machine \( i \) will be denoted by \( S_i: i.1, \ldots, i.j, \ldots, i.N_i \) where \( i.j \) is the job number of the \( j \)th job on machine \( i \) and \( N_i \) is the number of jobs assigned to machine \( i \). A schedule for \( K \) machines in parallel will be given by \( S: S_1; \ldots; S_j; \ldots; S_K \) where \( S_j \) is the schedule on machine \( j \) and \( \sum_{i=1}^{K} N_i = H \).

Two schedules will be said to be equal if the job sequences on each machine are identical. Since in this chapter only the one-machine case will be considered, this machine will be labeled \( 1 \) and \( N.1 = H \).

Lemma 3.1.—If for jobs numbered \( i \) and \( j \) in a schedule, \( f_i = f_j \),
d.i = d.j = t.0, a.i < a.j and jobs i and j are scheduled consecutively in a.i + a.j time units, i.e., if job number i = 1.k then job number j = l.k+1 or vice versa, then L(i,j) ≤ L(j.i) where L(i,j) is the loss if job i is scheduled before job j.

Proof.—Four cases will be used in the proof. Let c be the loss from all other jobs and let T be the time that the time interval (a.i + a.j) begins.

Case 1. — ∞ < t.0 ≤ T. This case is shown in Figure 2.

\[
\begin{align*}
&\quad |\quad b \quad | \quad a.i \quad | \quad a.j \quad | \\
&\quad | \quad | \quad | \\
&\quad t.0 \quad T
\end{align*}
\]

Figure 2.—Schedule

We then have

(3.1.1) \( L(i,j) = f.i(b + a.i) + f.j(b + a.i + a.j) + c \)

and

(3.1.2) \( L(j.i) = f.j(b + a.j) + f.i(b + a.i + a.j) + c \)

and the loss difference is

(3.1.3) \( L(j.i) - L(i,j) = f.j(b + a.i) - f.i(b + a.i) > 0. \)

Case 2. — T < t.0 ≤ T + a.i. This case is shown in Figure 3.

\[
\begin{align*}
&\quad |\quad a.i \quad | \quad a.j \quad | \\
&\quad | \quad | \quad |
\end{align*}
\]

Figure 3.—Schedule
We then have

\[(3.1.4) \quad L(i,j) = f_i(T + a_i - t.o) + f_j(T + a_i + a_j - t.o) + c\]

and

\[(3.1.5) \quad L(j,i) = f_j(T + a_j - t.o) + f_i(T + a_i + a_j - t.o) + c\]

and the loss difference is

\[(3.1.6) \quad L(j,i) - L(i,j) = f_x(T + a_j - t.o) - f_y(T + a_i - t.o) > 0.\]

**Case 3.** \(-T + a_i < t.o < T + a_j\). We then have

\[(3.1.7) \quad L(i,j) = f_j(T + a_i + a_j - t.o) + c\]

and

\[(3.1.8) \quad L(j,i) = f_j(T + a_j - t.o) + f_i(T + a_i + a_j - t.o) + c\]

and the loss difference is

\[(3.1.9) \quad L(j,i) - L(i,j) = f_x(T + a_j - t.o) > 0.\]

**Case 4.** \(-T + a_j \leq t.o < \infty\). We then have

\[(3.1.10) \quad L(i,j) = f_j(T + a_i + a_j - t.o) + c\]

and

\[(3.1.11) \quad L(j,i) = f_i(T + a_i + a_j - t.o) + c\]

and the loss difference is zero.

**Theorem 3.2.** If \(f_1 = \ldots = f_H, d_1 = \ldots = d_H = t.o \leq 0\), and the jobs are numbered so that \(a_i \leq \ldots \leq a_j \leq \ldots \leq a_H\), then the set of optimal schedules consists of \(S: 1, 2, \ldots, H\) and schedules obtained by starting with \(S\) and interchanging jobs \(i\) and \(j\) whenever \(a_i = a_j\).

**Proof.** Let \(S' : 1, 1, \ldots, 1, j, \ldots, 1, H\) be any schedule not in the set defined in the theorem. Then there exists a smallest job number
in $S^*$, say job number $j'$, such that $1.j' > j'$ and $a.1.j' > a.j'$. Find job $j'$ and determine its position in $S^*$, call this position $h$. Since $j'$ was chosen to be the smallest, we know that $a.j' = a.1.h < a.1.h-l$. Therefore, by Case 1 of Lemma 3.1 a better schedule is $S''$: $1.1, \ldots, 1.h, 1.h-l, \ldots, 1.H$ so $S'$ is not optimal.

Since the set of schedules defined in the theorem contains only $S: 1, 2, \ldots, H$ and schedules obtained by interchanging jobs $i$ and $j$ in $S$ when $a.i = a.j$, every schedule in the set has the same loss. Since there are only a finite number of schedules and thus at least one optimal schedule, all schedules in the set defined in the theorem must be optimal; otherwise, there would be no optimal schedule.

**Theorem 3.3.**—If $f.1 = \ldots = f.H$, $d.1 = \ldots = d.H = t.0 > 0$, and the jobs are numbered so that $a.1 \leq \ldots \leq a.H$, then an optimal schedule is $S: 1, 2, \ldots, H$.

**Proof.**—If $t.0 \geq a.1 + \ldots + a.H$, the theorem follows immediately. If $t.0 < a.1 + \ldots + a.H$, let $S'$ be any schedule. It will be shown that $S'$ may be transformed into $S$ by interchanging jobs without increasing the loss. It then follows that $S$ is optimal.

If $S' \neq S$, there exists a smallest job number, $j'$, in $S'$ such that job number $1.j' > j'$. Find $j'$. Since $j'$ was chosen as the smallest job number such that $1.j' > j'$, we know that $a.j' = \min(a.1.j', a.1.j'+1, \ldots, a.1.H)$. Therefore, by repeated application of Lemma 3.1 we can interchange adjacent jobs without increasing the loss until job number

---

1Note that $a.1.j'$ is the total operation time for the $j'$th job on machine 1 and not the $j'$th part of job 1 as in Chapter II. If 1.5 is job number 3, then $a.1.5 = a.3$. 


1. $j' = j'$. By induction it follows that we can do this for all jobs.

One might think that $S$ and schedules obtained by arranging in any order those jobs in $S$ whose completion times are less than or equal to $t_0$ were the only optimal schedules. The following example shows this not to be the case. Let $a_i = i$, $i = 1, \ldots, 5$, and $t_0 = 7$. An optimal schedule is $S': 1, 2, 4, 3, 5$ and the loss is $f_3(3) + f_5(8)$. The schedule $S: 1, \ldots, 5$ yields an equivalent loss of $f_4(3) + f_5(8)$.

Lemma 3.4.—Let $f'(x)$, the derivative of $f(x)$, be nondecreasing and non-negative for $x \geq 0$ and let $a \geq b \geq 0$, $c \geq d \geq 0$, $a \geq c$, and $a - b \geq c - d$, then $[f(a) - f(b)] - [f(c) - f(d)] \geq 0$.

Proof.—Let $b'$ be such that $a - b' = c - d = w$. Then $b' \geq b$ and $f(b') \geq f(b)$. If $f'(x) = 0$, then $f(a) - f(b') = f(c) - f(d)$. Let $f'(x) > 0$. Since $b' \geq d$, $f(w + b') - f(w + d) \geq f(b') - f(d)$ and $f(a) - f(b) \geq f(a) - f(b') \geq f(c) - f(d)$.

Theorem 3.5.—Number the jobs so that $a_1 \leq \ldots \leq a_N$ and so that if $a_i = a_{i+1}$, then $d_i \leq d_{i+1}$. If $f_1 = \ldots = f_N$, $f_1$ satisfies the conditions of Lemma 3.4, $d_1 \leq d_2 \leq \ldots \leq d_N$ and $d_N \leq d_i$ for all $i > N$, then an optimal schedule will begin with jobs 1, 2, $\ldots$, $N$.

Proof.—Let $S: 1, 1, \ldots, 1$ be any schedule not satisfying the conclusion of the theorem. Then there exists a smallest job number in $S$, say $j'$, such that $1,j' > j' \leq N$. Then there also exists an $h > j'$ such that $1,h = j'$. Consider jobs $1,h$ and $1,h-1$ and call them jobs 1 and 2 respectively. Since $j'$ was chosen to be the smallest job number such that $1,j' > j' \leq N$, we know that $a_1 \leq a_2$ and $d_1 \leq d_2$. 
Let $c$ be the loss from all other jobs. This schedule is shown in Figure 4.

![Figure 4.—Schedule](image)

We then have

\[(3.5.1) \quad L(2,1) = f_2(T + a_2 - d_2) + f_1(T + a_1 + a_2 - d_1) + c\]

and

\[(3.5.2) \quad L(1,2) = f_2(T + a_1 + a_2 - d_2) + f_1(T + a_1 - d_1) + c\]

and the loss difference is

\[(3.5.3) \quad L(2,1) - L(1,2) = \left[ f_1(T + a_1 + a_2 - d_1) - f_1(T + a_1 - d_1) \right] - \left[ f_2(T + a_1 + a_2 - d_2) - f_2(T + a_2 - d_2) \right].\]

By Lemma 3.4, \((3.5.3) \geq 0\) and we can interchange jobs $l.h$ and $l.h-1$ without increasing the loss. Equation \((3.5.3)\) is independent of the location of $d_1$ and $d_2$ in Figure 4. Interchange jobs $l.h$ and $l.h-1$ and repeat this procedure until schedule $S$ begins with jobs $1, 2, \ldots, N$.

**Theorem 3.6.**—Number the jobs so that $a_1 \leq \ldots \leq a_H$ and so that if $a_i = a_{i+1}$, then $d_i \leq d_{i+1}$. If $f_1 = \ldots = f_H$, $f_1$ satisfies the conditions of Lemma 3.4, $d_M \leq \ldots \leq d_H$ and $d_M \geq d_i$ for all $i < M$, then an optimal schedule will end with jobs $M$, $M + 1$, $\ldots$, $H$.

**Proof.**—Let $S: 1.1, \ldots, l.H$ be any schedule not satisfying the conclusion of the theorem. Then there exists a largest job number in $S$, \[\ldots, H.\]
say \( j' \), such that \( 1 \cdot j' < j' \geq M \). There also exists an \( h < j' \) such that \( 1 \cdot h = j' \). Consider jobs \( 1 \cdot h \) and \( 1 \cdot h+1 \) and call them jobs 2 and 1 respectively. Since \( j' \) was chosen to be the largest job number such that \( 1 \cdot j' < j' \geq M \), we know that \( a.1 \leq a.2 \) and \( d.1 \leq d.2 \). At this point the proof continues as in Theorem 3.5 from which it follows that we may interchange jobs \( 1 \cdot h \) and \( 1 \cdot h+1 \) without increasing the loss. This procedure may be repeated until \( S \) ends with jobs \( M, M+1, \ldots, H \).

As an example in the use of Theorems 3.5 and 3.6, suppose \( a.1 = 1 \), \( i = 1, \ldots, 8 \) and \( d.1 = 1, d.2 = 1, d.3 = 2, d.4 = 4, d.5 = 4, d.6 = 7, d.7 = 5 \), and \( d.8 = 7 \). From Theorem 3.5 the schedule will begin with jobs \( 1, 2, \ldots, 5 \) and from Theorem 3.6 the schedule will end with job 8 and it only remains to determine if job 6 should precede job 7 or vice versa.

If a company took orders using a policy that the larger the order the greater the promised delivery date, Theorem 3.5 would give them a schedule rule. Theorems 3.5 and 3.6 may also be used in conjunction with the general method of Chapter II. If from Theorem 3.5 we can place \( N \) jobs and from Theorem 3.6 \( M \) jobs, then we have \( H-N-M \) jobs to be scheduled and the number of calculations in the general method will be \( 2^{H-N-M-1} \).

Most of the theorems have indicated that the jobs with the shortest operation times should be scheduled at the beginning of the sequence. If the deadlines for the jobs are all the same, this has been proved to be true; however, if the deadlines vary, this will probably not be the case in most situations.
For example, if \( a_i = i = 1, 2, 3 \), \( d_1 = 3, d_2 = 0, d_3 = 3 \), and \( f_i(x) = x \), an optimal job sequence is 2, 1, 3, with a loss of five. Scheduling the shortest jobs first yields a loss of six. One may wonder, however, if using this "rule of thumb" might not lead to a fairly "good" schedule. The following theorem will give us an upper bound on the error we might make using this "rule of thumb".

**Theorem 3.7.**—Let \( d^* = \max_i d_i; a_1 \leq \ldots \leq a_H; S^*: 1.2.\ldots .H \); \( L(i, d, S) \) denote the loss from job \( i \) in schedule \( S \) when \( d_i = d \) and let \( S' \) be any optimal schedule. If \( f_1 = \ldots = f_H \), then

\[
(3.7.1) \quad \sum_{i=1}^{H} L(i, d_i, S^*) - \sum_{i=1}^{H} L(i, d_i, S') \leq \sum_{i=1}^{H} L(i, d_i, S^*) - \sum_{i=1}^{H} L(i, d^*, S^*).
\]

**Proof.**—Since increasing the deadline for job \( i \) from \( d_i \) to \( d^* \) will not increase the loss from job \( i \) in any schedule, we have

\[
(3.7.2) \quad L(i, d^*, S) \leq L(i, d_i, S).
\]

From (3.7.2) it follows that

\[
(3.7.3) \quad \sum_{i=1}^{H} L(i, d^*, S') \leq \sum_{i=1}^{H} L(i, d_i, S').
\]

When \( d_i = d^* \) for all \( i \), we know from Theorems 3.2 and 3.3 that an optimal schedule is \( S^* \) and it then follows that

\[
(3.7.4) \quad \sum_{i=1}^{H} L(i, d^*, S^*) \leq \sum_{i=1}^{H} L(i, d^*, S').
\]

Equation (3.7.1) follows from (3.7.3) and (3.7.4).
CHAPTER IV
THEOREMS FOR H JOBS ON K MACHINES

Using methods similar to those employed in Chapter III, this chapter will explore the problem of scheduling H jobs on K machines in parallel. The information developed in Chapter III will be frequently used in this chapter.

Theorem 4.1.—Let there be K identical machines and let N.k be the number of jobs assigned to machine k. If $f_i(x) = bx + c_i$, $b > 0$, $c_i \geq 0$ and $d.1 = \ldots = d.H = t.0 \leq 0$, then any optimal schedule will have $|N.i - N.j| \leq 1$ for all machines i and j.

Proof.—It is sufficient to consider only the case $t.0 = 0$ since if $d.i < 0$, we may incorporate the loss $b|d.i|$ into $c.i$ and make $d.i = 0$.

Suppose there is an optimal schedule with machines i and j such that $|N.i - N.j| > 1$. We may assume without loss of generality that $N.i - N.j > 1$. Let the jobs on machine i be $i.1, \ldots, i.N.i$ and the jobs on machine j be $j.1, \ldots, j.N.j$. Since $t.0 = 0$ and thus all jobs are late, the total loss contributed by the $c.i$ may be neglected. The loss on machine i is

\[
L(i.1, \ldots, i.N.i) = \sum_{h=1}^{N.i} b(a.i.1 + \ldots + a.i.h)
\]

\[
= b[(N.i)a.i.1 + (-1 + N.i)a.i.2 + \ldots + a.i.N.i].
\]

The loss on machine j is
(4.1.2) \[ L(j.1, \ldots, j.N.j) = \sum_{h=1}^{N_{j}} b(a_{j.1} + \ldots + a_{j.h}) \]
\[ = b[(N_{j})a_{j.1} + (-1 + N_{j})a_{j.2} + \ldots + a_{j.N.j}] \]

If job i.1 is put first on machine j, the loss on machine j is

(4.1.3) \[ L(i.1l, j.1, \ldots, j.N.j) = b[a_{i.1} + \sum_{h=1}^{N_{j}} (a_{i.1} + a_{j.1} + \ldots + a_{j.h})] = b[(1 + N_{j})a_{i.1} + (N_{j})a_{j.1} + \ldots + a_{j.N.j}] \]

and the loss on machine i is

(4.1.4) \[ L(i.2, \ldots, i.N.i) = \sum_{h=2}^{N_{i}} b(a_{i.2} + \ldots + a_{i.h}) \]
\[ = b[(-1 + N_{i})a_{i.2} + \ldots + a_{i.N.i}] \]

The loss for the original schedule less the loss for the revised schedule is

(4.1.5) \[ D = b[(N_{i})a_{i.1} - (1 + N_{j})a_{i.1}] = b[a_{i.1}((N_{i}) - (N_{j}) - 1)] \]

Since \( a_{i.1} > 0 \) and \( N_{i} - N_{j} > 1 \), \( D > 0 \).

**Theorem 4.2.**—If the hypothesis of Theorem 4.1 is satisfied, then there exists an optimal schedule satisfying

(4.2.1) \( N_{i} \geq N_{2} \geq \ldots \geq N_{K} \) and \( N_{1} - N_{K} \leq 1 \),

(4.2.2) \( a_{i.1} \leq a_{i.2} \leq \ldots \leq a_{i.N.i}, i = 1, \ldots, K \),

and

(4.2.3) \( a_{i.j} \leq a_{2.j} \leq \ldots \leq a_{h.j} \) for all \( j \) (for a given \( j \), \( h \) equals the largest number such that machine \( h \) has a \( j \)th job)

where \( a_{i.j} \) is the operation time for the \( j \)th job on machine \( i \).

**Proof.**—As in Theorem 4.1, we may assume \( t_0 = 0 \). By Theorems 3.2 and 4.1 there exists an optimal schedule satisfying conditions (4.2.1) and (4.2.2). Suppose a schedule satisfied (4.2.1) and (4.2.2) but not
(4.2.3) This implies that there exists and we can choose a smallest position number, say \( j \), and for that \( j \) a smallest machine number, say \( x \), such that \( a.x.j > a.y.j \) for some \( y > x \). We may choose \( y \) such that \( a.y.j \leq a.h.j \), \( x < h \leq K \). Since all jobs are late, the loss contributed by the \( c.i \) may be neglected.

The loss for this schedule on machine \( x \) is

\[
(4.2.4) \quad L(x.1, \ldots, x.N.x) = b[(N.x)a.x.1 + (-1 + N.x)a.x.2 + \ldots + (1-j + N.x)a.x.j + \ldots + a.x.N.x].
\]

The loss for this schedule on machine \( y \) is

\[
(4.2.5) \quad L(y.1, \ldots, y.N.y) = b[(N.y)a.y.1 + (-1 + N.y)a.y.2 + \ldots + (1-j + N.y)a.y.j + \ldots + a.y.N.y].
\]

If jobs \( x.j \) and \( y.j \) are interchanged, the loss on machine \( x \) is

\[
(4.2.6) \quad L(x.1, \ldots, x.j-1, y.j, x.j+1, \ldots, x.N.x) = L(x.1, \ldots, x.N.x) + b(1 - j + N.x)(a.y.j - a.x.j)
\]

and the loss on machine \( y \) is

\[
(4.2.7) \quad L(y.1, \ldots, y.j-1, x.j, y.j+1, \ldots, y.N.y) = L(y.1, \ldots, y.N.y) + b(1 - j + N.y)(a.x.j - a.y.j).
\]

The loss for the original schedule less the loss for the revised schedule is

\[
(4.2.8) \quad D = b[(1 - j + N.x)(a.x.j - a.y.j) - (1 - j + N.y)(a.x.j - a.y.j)]
\]

\[
= b(a.x.j - a.y.j)((N.x) - (N.y)) \geq 0.
\]

It is possible that in interchanging jobs \( x.j \) and \( y.j \) condition (4.2.2) may not hold.\(^1\) Since \( a.y.j-1 \geq a.x.j-1 \), \( a.y.j \geq a.y.j-1 \),

\(^1\)For example, if \( K = 3 \) and the sequence of operation times on machine 1 is 1, 4, 5, on machine 2 is 2, 3, 3 and on machine 3 is 3, 4, 4 and jobs 1.2 and 2.2 are interchanged, condition (4.2.2) will not hold on machine 2.
and \( a.x.j+1 \geq a.x.j \), we have \( a.x.j-1 \leq a.y.j \leq a.x.j+1 \) and condition (4.2.2) holds on machine \( x \). Now \( a.x.j > a.y.j-1 \) and in fact \( a.x.j \) may be greater than \( a.y.j+h \), \( 1 \leq h \leq (N.y) - j \). If this is the case put the jobs on machine \( y \) into the order satisfying condition (4.2.2). By Theorem 3.2 this will not increase the loss. We now have a schedule satisfying conditions (4.2.1), (4.2.2) and condition (4.2.3) through the \( j \)th job on machine \( x \). We may now repeat this procedure until condition (4.2.3) is completely satisfied.

It is to be noted that this theorem does not hold if \( f(x) \) is not linear. Consider, for example, the case when \( f.i(x) = x^2 \), \( a.1 = 1 \), \( a.2 = 1 \), \( a.3 = 1 \), and \( a.4 = 50 \) and there are two machines. If the schedule is \( S: 1, 2; 3, 4 \), the loss is 2607, but if the schedule is \( S: 1, 2, 3; 4 \), the loss is 2514.

**Theorem 4.3.**—Let there be \( K \) identical machines and \( a.1 \leq \ldots \leq a.H \). If \( f.i(x) = bx + c.i \), \( b > 0 \), \( c.i \geq 0 \) and \( d.1 = \ldots = d.H = t.0 \leq 0 \), then an optimal schedule, \( S^* \), is obtained using the formula

\[
(4.3.1) \quad p.q = (q - 1)K + p, \quad p = 1, \ldots, K; \quad q = 1, \ldots, N.p
\]

where \( N.p \) is the largest integer such that \( p.N.p \leq H \), and \( p.q \) is the job number of the \( q \)th job on machine \( p \).

**Proof.**—As in Theorem 4.1 it is sufficient to consider only the case \( t.0 = 0 \). The class of schedules defined in Theorem 4.2 contains an optimal schedule. It will be shown that any schedule in this class, including an optimal schedule, can be transformed without increasing the loss into \( S^* \) and thus the theorem will be proved.

Let \( S \) be any schedule in the set of schedules defined in Theorem
4.2 which does not satisfy the formula (4.3.1)\(^1\) for some jobs on one or more machines. Then there exists in \(S\) a smallest position number, say \(j'\), and for that position number a smallest machine number, say \(i'\), such that job number \(i'.j' > (j' - 1)K + i' = w\). Then job number \(w\) must be in some other place in the schedule \(S\). Let job number \(w\) be on the \(i''\)th machine in the \(j''\)th position. Since in \(S^*\) job 1.1 = 1, 2.1 = 3, \ldots, \(K.1 = K, 1.2 = K + 1, \ldots, K.2 = 2K, \ldots, 1.3 = 2K + 1, \ldots, a.1 \leq \ldots \leq a.N\), and job number \(w\) is the smallest job number in \(S\) not in its position given by the formula (4.3.1), \(a.1'.j > a.w = a.i''.j''\) and \(j'' > j'\).

Since \(S\) is in the set defined by Theorem 4.2, \(j'' > j'\) for if \(j'' = j'\), then \(i'' = i'\) by condition (4.2.3) of Theorem 4.2 and the assumption that \(i'.j' \neq i''.j''\) is contradicted.

The loss for schedule \(S\) on machine \(i'\) is

\[
L(i'.1, \ldots, i'.N.i') = \sum_{h=1}^{N.i'} b(a.i'.1 + \ldots + a.i'.h)
\]

\[
= b[(N.i')a.i'.1 + \ldots + (1 - j' + N.i')a.i'.j' + \ldots + a.i'.N.i']
\]

Since all jobs are late, the loss contributed by the \(c.i\) may be neglected. The loss on machine \(i''\) is

\[
L(i''.1, \ldots, i''.N.i'') = \sum_{h=1}^{N.i''} b(a.i''.1 + \ldots + a.i''.h)
\]

\[
= b[(N.i'')a.i''.1 + \ldots + (1 - j'' + N.i'')a.i''.j'' + \ldots + a.i''.N.i'']
\]

\(^1\)For example, if \(a.i = i\), \(i = 1, 2, 3\) and \(K = 2\), the schedule \(S: 1, 2; 3\) is an optimal schedule which is in the set defined by Theorem 4.2, but which does not satisfy the formula of Theorem 4.3.
If jobs $i'.j'$ and $i''.j''$ are interchanged, the loss on machine $i'$ is
\begin{equation}
L(i'.1, \ldots, i'.j'-1, i''.j'', i'.j'+1, \ldots, i'.N.i')
\end{equation}
\begin{equation}
= L(i'.1, \ldots, i'.N.i') + b(1 - j' + N.i')(a.i''.j'' - a.i'.j')
\end{equation}
and the loss on machine $i''$ is
\begin{equation}
L(i''.1, \ldots, i''.j''-1, i'.j', i''.j'+1, \ldots, i''.N.i'')
\end{equation}
\begin{equation}
= L(i''.1, \ldots, i''.N.i'') + b(1 - j'' + N.i'')(a.i'.j' - a.i''.j'').
\end{equation}

The loss for the original schedule less the loss for the revised schedule is
\begin{equation}
D = b[(1 - j' + N.i')(a.i''.j'' - a.i'.j')
\end{equation}
\begin{equation}
+ (1 - j'' + N.i'')(a.i'.j' - a.i''.j'')]
\end{equation}
\begin{equation}
= b[a.i''.j'' - a.i''.j''][(N.i' - N.i'') + (j'' - j')].
\end{equation}

Since $j'' - j' \geq 1$ and $|N.i' - N.i''| \leq 1$, $D \geq 0$.

It is possible that after interchanging jobs $i'.j'$ and $i''.j''$, conditions (4.2.2) and (4.2.3) of Theorem 4.2 may not hold on machines $i'$ and $i''$ and for the $j'$th and $j''$th jobs. 1 Since $w = i''.j''$ was the smallest job number not assigned by the formula (4.3.1) and $a.i'.j' \geq a.i''.j''$, we have $a.i'.j'-1 \leq a.i''.j'' \leq a.i'.j'+1$ and $a.i'-1.j' \leq a.i''.j'' \leq a.i'+1.j'$. It follows that condition (4.2.2) is satisfied for machine $i'$ and condition (4.2.3) for the $j'$th job on each machine. Now $a.i'.j' \geq a.i''.j''-1$ and in fact $a.i'.j'$ may be greater than $a.i''.j''+h$, $1 \leq h \leq (N.i'') - j''$. If this is the case by Theorem 3.2 we can

---

1For example, let the sequence of operation times on machine 1 be 1, 2, 2, on machine 2 be 2, 2, 2, and on machine 3 be 3, 3, 3, and then interchange jobs 3.1 and 1.2.
arrange jobs \( i^*.j^* \), \( i^*.N.i^* \) without increasing the loss so that condition (4.2.2) is satisfied on machine \( i^* \). We may now apply Theorem 4.2 (actually the procedure in the proof) and satisfy condition (4.2.3) for the \( j^* + 1, j^* + 2, \ldots \) jobs on all machines without increasing the loss. At this point the entire procedure may be repeated until \( S = S^* \). Since \( S^* \) is unique and at least one schedule in the set defined by Theorem 4.2 is optimal, \( S^* \) must be optimal.

For example, if we have jobs \( a.i = i, i = 1, \ldots, N \), and two machines, an optimal schedule is obtained by putting in ascending order the odd number jobs on one machine and the even number jobs on the other machine. This example seems to go against one's intuition since the difference in the operating times for the two machines increases with \( N \).

**Theorem 4.4.**—Let there be \( K \) identical machines and let \( N^*.k \) be the number of jobs assigned to machine \( k \) whose starting times are \( \geq t.0 \).

If \( f.1 = \ldots = f.H = bx + c.i, b > 0, c.1 \geq 0, d.1 = \ldots = d.H = t.0 > 0 \), then any optimal schedule will have \( |N^*.i - N^*.j| \leq 1 \) for all machines \( i \) and \( j \).

**Proof.**—Assume there exists an optimal schedule \( S \) with any pair of machines \( i \) and \( j \) such that \( |N^*.i - N^*.j| > 1 \). We may assume without loss of generality that \( N^*.i - N^*.j > 1 \). For machine \( p \) let job \( p.p^* \) be the last job on machine \( p \) whose starting time is \( < t.0 \) and let job \( p.p^* \) end \( P \) time units, \( 0 \leq P \leq a.p.p^* \), after \( t.0 \). Then the jobs on machine \( p \) which may contribute to the loss are \( p.p^*, p.p^*+1, \ldots, p.p^*+N^*.p \).  

\[ 1 \text{In the remaining theorems } p^* \text{ in } p.p^*+h, 0 \leq h \leq N^*.p, \text{ will often be dropped when in the proofs we are only going to be concerned with those jobs whose completion times are } \geq t.0. \text{ In these cases we will simply write } p.0, p.1, \ldots, p.N^*.p \text{ for } p.p^*+0, p.p^*+1, \ldots, p.p^*+N^*.p \text{ as in equation (4.4.1).} \]
We will assume that \( c_i = 0, \ i = 1, \ldots, H \) and later show that this assumption was not necessary.

The loss on machine \( i \) is

\[
(4.4.1) \quad L(i, 0, i, 1, \ldots, i, N^*, i) = \sum_{h=0}^{N^* - 1} b(I + a_i.i + \ldots + a_i.h) \\
= b[(1 + N^* - 1)I + (N^* - 1)a_i.1 + \ldots + a_i.N^*].
\]

The loss on machine \( j \) is

\[
(4.4.2) \quad L(j, 0, j, 1, \ldots, j, N^*, j) = \sum_{h=0}^{N^* - 1} b(J + a_j.j + \ldots + a_j.h) \\
= b[(1 + N^* - 1)J + (N^* - 1)a_j.1 + \ldots + a_j.N^*].
\]

**Case 1.** \( a_i.1 > J \geq 0 \). If job \( i, 1 \) is placed between \( j, 0 \) and \( j, 1 \) on machine \( j \), the situation is shown in Figure 5.

![Figure 5. Schedule](image)

The loss on machine \( j \) is

\[
(4.4.3) \quad L(j, 0, i, 1, j, 1, \ldots, j, N^*, j) \\
= bJ + b(J + a_i.1) + \sum_{h=1}^{N^* - 1} b(J + a_i.1 + a_j.j + \ldots + a_j.h)
\]

Several cases will be used in this proof and in the proofs of other theorems. This is necessitated by the fact that, depending on the operation times, some job interchanges may make an optimal schedule non-optimal.

and the loss on machine \(i\) is

\[(4.4.4) \quad L(i.0, i.2, \ldots, i.N^*.i) = bI + \sum_{h=2}^{N^*.i} b(I + a.i.2 + \ldots + a.i.h) \]

\[= b[(N^*.i)I + (-1 + N^*.i)a.i.2 + \ldots + a.i.N^*.i]. \]

The loss for the original schedule less the loss for the revised schedule is

\[(4.4.5) \quad D = b[I + a.i.1(N^*.i) - a.i.1(1 + N^*.j) - J] \]

\[= b[(I - J) + ((N^*.i) - (N^*.j) - 1)a.i.1] \geq b(a.i.1 - J) > 0. \]

This contradicts the contrary hypothesis that \(S\) was optimal.

**Case 2.** \(0 < a.i.1 \leq J\). If job \(i.1\) is placed before \(j.0\) on machine \(j\), the situation is shown in Figure 6. \(J.1\) is the number of late units for job \(i.1\) on machine \(j\). \(J.1\) could be negative.

![Figure 6.—Schedule](image)
Case 2.1. \(-J.l \geq 0\). The loss on machine \(j\) is

\[
L(i.1, j.0, \ldots, j.N*.j) = bJ.l + \sum_{h=0}^{N*.j} b(J.l + a.j.0 + \ldots + a.j.h)
\]

\[
= b[(2 + N*.j)J.l + (1 + N*.j)a.j.0 + \ldots + a.j.N*.j].
\]

The loss for the original schedule less the loss for the revised schedule is \((4.4.1) + (4.4.2) - (4.4.4) - (4.4.6)\) which gives

\[
D = b[I + (N*.i)a.i.1 + (1 + N*.j)J.l - (2 + N*.j)J.l - (1 + N*.j) a.j.0].
\]

![Diagram of Schedule on Machine j](image)

From Figure 7 we see that

\[
A = a.j.0 - a.i.1
\]

and

\[
J.l = J - A = J - a.j.0 + a.i.1.
\]

Substituting \((4.4.9)\) in \((4.4.7)\) we have

\[
D = b[I + ((N*.i) - (N*.j) - 2)a.i.1 - J + a.j.0] \geq b(a.j.0 - J)
\]

\[
> 0.
\]

This contradicts the contrary hypothesis that \(S\) was optimal.
Case 2.2. -- \( J \cdot 1 < 0 \). The loss on machine \( j \) is

\[
L(j, 0, \ldots, j, N^*j) = \sum_{h=0}^{N^*j} b[(J \cdot 1 + a \cdot j \cdot 0) + a \cdot j \cdot l + \ldots + a \cdot j \cdot h]
\]

\[
= b[(1 + N^*j)(J \cdot 1 + a \cdot j \cdot 0) + (N^*j)a \cdot j \cdot l + \ldots + a \cdot j \cdot N^*j].
\]

The loss for the original schedule less the loss for the revised

\[
D = b[I + (N^*j)a \cdot i \cdot l + (1 + N^*j)(J \cdot 1 - a \cdot j \cdot 0)]
\]

\[
> b[(1 + N^*j)(J - J \cdot 1 - a \cdot j \cdot 0 + a \cdot i \cdot l)].
\]

Equations (4.4.8) and (4.4.9) still hold if \( J \cdot 1 < 0 \) and we have

(4.4.13) \( J - J \cdot 1 = a \cdot j \cdot 0 - a \cdot i \cdot l \).

Substituting (4.4.13) into (4.4.12) yields \( D > 0 \). This contradicts the contrary hypothesis that \( S \) was optimal.

The assumption that the \( c \cdot i \) were all zero was not necessary. An examination of the three cases shows that the late jobs in the revised schedules form a subset of the late jobs in the original schedule and thus including the loss from the \( c \cdot i \) will not increase the loss difference between the original schedule and the revised schedules.

Theorem 4.5. -- If \( a \cdot 1 \leq \ldots \leq a \cdot H, f \cdot 1 = \ldots = f \cdot H = bx, b > 0, \)
and \( d \cdot 1 = \ldots = d \cdot H = t \cdot 0 > 0 \), then there exists an optimal schedule such that, for any pair of jobs \( i \) and \( j \) with \( a \cdot i > a \cdot j \), if job \( i \) is started before the common deadline \( t \cdot 0 \) then job \( j \) is started before time \( t \cdot 0 \).

---

1For example, if \( a \cdot 1 = 1, i = 1, \ldots, 5, K = 2 \) and \( t \cdot 0 = 3 \), the schedule \( S: 1, 2, 4; 3, 5 \) is an optimal schedule and satisfies Theorem 4.5.
Proof.—There exists an optimal schedule such that Theorem 3.3 is satisfied for each individual machine. It will be shown by contradiction that there must exist at least one optimal schedule in the set of optimal schedules satisfying Theorem 3.3 which also satisfies Theorem 4.5.

Consider any optimal schedule which satisfies Theorem 3.3 but which does not satisfy Theorem 4.5 and suppose there are just two jobs, say jobs $i'$ and $j'$, which violate Theorem 4.5. Since the schedule satisfies Theorem 3.3 the two jobs are not on the same machine.

The proof will follow in two parts. The first part will develop three types of job interchanges which will not increase the loss of the schedule. The second part will be a proof by contradiction which will utilize the three types of job interchanges developed in the first part.

Since there are just two jobs violating Theorem 4.5 let the jobs be on machines $i$ and $j$ and suppose machine $i$ has a job, say $i'$, which starts before $t.0$ and machine $j$ has a job, say $j'$, which starts on or after $t.0$ and $a.i' > a.j'$. Using the convention described in Theorem 4.4 job $i.0$ is the last job on machine $i$ which starts before $t.0$ and job $j.1$ is the first job on machine $j$ which starts on or after $t.0$. Since Theorem 3.3 is satisfied on both machines, the operation time for job $i'$ is less than or equal to the operation time for job $i.0$ and the operation time for job $j'$ is greater than or equal to the operation time for job $j.1$. Therefore, $a.i.0 \geq a.i' > a.j' \geq a.j.1$ and the two jobs which violate Theorem 4.5 must be $i.0$ and $j.1$. This schedule is shown in Figure 8.
The loss on machine $i$ is

\[(4.5.1) \quad L(i, 0, \ldots, i.N^*.i) = \sum_{h=0}^{N^*.i} b(I + a.i.l + \ldots + a.i.h)\]

\[= b[(1 + N^*.i)I + (N^*.i)a.i.l + \ldots + a.i.N^*.i].\]

The loss on machine $j$ is

\[(4.5.2) \quad L(j, 0, \ldots, j.N^*.j) = \sum_{h=0}^{N^*.j} b(J + a.j.l + \ldots + a.j.h)\]


Case 1. $a.j.l \geq a.i.0 - I$. If jobs $i.0$ and $j.1$ are interchanged, the schedule appears as in Figure 9.

Figure 8.—Schedule

Figure 9.—Schedule
Since $J \geq 0$ and since $a.j.l \geq a.i.0 - I, I.l \geq 0$. The loss on machine $i$ is

\begin{equation}
(4.5.3) \quad L(j.l, i.l, \ldots, i.N*.i) = \sum_{h=0}^{N*.i} b[I.l + a.i.l + \ldots + a.i.h] \\
= b[(1 + N*.i)I.l + (N*.i)a.i.l + \ldots + a.i.N*.i].
\end{equation}

The loss on machine $j$ is

\begin{equation}
(4.5.4) \quad L(j.0, i.0, j.2, \ldots, j.N*.j) = bJ + b(J - a.i.0) \\
+ \sum_{h=2}^{N*.j} b(J + a.i.0 + a.j.2 + \ldots + a.j.h) \\
= b[(1 + N*.j)J + (N*.j)a.i.0 + (-1 + N*.j)a.j.2 + \ldots \\
+ a.j.N*.j].
\end{equation}

The loss for the original schedule less the loss for the revised schedule is $(4.5.1) + (4.5.2) - (4.5.3) - (4.5.4)$ which gives

\begin{equation}
(4.5.5) \quad D = b[(1 + N*.i)(I - I.l) + (N*.j)(a.j.l - a.i.0)] \\
= b[(1 + N*.i)(I - I.l) + (N*.j)(a.j.l - a.i.0)].
\end{equation}

Noting that

\begin{equation}
(4.5.6) \quad I - I.l = a.i.0 - a.j.l
\end{equation}

and substituting (4.5.6) in (4.5.5) we have

\begin{equation}
(4.5.7) \quad D = b[(a.i.0 - a.j.l)(1 + N*.i - N*.j)].
\end{equation}

By Theorem 4.4 and the fact that $a.i.0 - a.j.l > 0$, $D \geq 0$.

**Case 2.1.** $a.j.l < a.i.0 - I$, $N*.i \geq N*.j$. If jobs $i.0$ and $j.1$ are interchanged, the situation is the same as Case 1 except $I.l < 0$. The loss on machine $i$ is

\begin{equation}
(4.5.8) \quad L(i.l, \ldots, i.N*.i) = \sum_{h=1}^{N*.i} b[I.l + a.i.l + \ldots + a.i.h]
\end{equation}
= b[(N*.i)I.l + (N*.i)a.i.l + ... + a.i.N*.i]

The loss for the original schedule less the loss for the revised schedule is (4.5.1) + (4.5.2) - (4.5.4) - (4.5.8) which gives

(4.5.9) \( D = b[(1 + N*.i)I - (N*.i)I.l + N*.j(a.j.l - a.i.0)] \)

\[ \geq b(N*.i(I - I.l) + N*.j(a.j.l - a.i.0)]. \]

Equation (4.5.6) also holds for I.1 < 0. Substituting (4.5.6) in (4.5.9) we have

(4.5.10) \( D \geq b(a.i.0 - a.j.1)(N*.i - N*.j) \geq 0. \)

Case 2.2.—a.j.1 < a.i.0 - I, N*.i < N*.j. In this case take the first job on machine j and put it first on machine i. Suppose that the first job on machine j was not j.0 and that after transferring the first job from machine j to machine i, j.0 is still the last job whose starting time is < t.0. This situation is shown in Figure 10.

\[ \text{Figure 10.—Schedule} \]

The loss equations for machines i and j are the same as (4.5.1) and (4.5.2) except I' replaces I and J' replaces J. The loss for the original schedule less the loss for the revised schedule is

(4.5.11) \( D = b[(1 + N*.i)(I - I') + (1 + N*.j)(J - J')]. \)
Noting that

\[(4.5.12) \quad I' - I = J - J' > 0\]

and substituting (4.5.12) in (4.5.11), we have

\[(4.5.13) \quad D = (J - J')(N*.j - N*.i) > 0.\]

Since the original schedule was optimal the case shown in Figure 10 cannot occur.

Therefore, the first job on machine \(j\) is \(j.0\) or after transferring the first job from machine \(j\), \(j.0\) is not the last job whose starting time is \(< t.0\). Then after the transfer of Case 2.2, the situation is as shown in Figure 11.

![Schedule Diagram](https://example.com/schedule.png)

**Figure 11. Schedule**

The loss on machine \(i\) is

\[(4.5.14) \quad L(i.0, \ldots, i.N*.i) = \sum_{h=0}^{N*.i} b(I' + a.i.1 + \ldots + a.i.h)\]

\[= b[(1 + N*.i)I' + (N*.i)a.i.1 + \ldots + a.i.N*.i].\]

The loss on machine \(j\) is

\[(4.5.15) \quad L(j.1, \ldots, j.N*.j) = \sum_{h=1}^{N*.j} b(J^* + a.j.2 + \ldots + a.j.h)\]

\[= b[(N*.j)J^* + (-1 + N*.j)a.j.2 + \ldots + a.j.N*.j].\]
The loss for the original schedule less the loss for the revised schedule is (4.5.1) + (4.5.2) - (4.5.14) - (4.5.15) which gives

\[(4.5.16) \quad D = b[(1 + N*.i)(I - I') + (1 + N*.j)J + N*.j(a.j.l - J^m)].\]

Let the operation time for the job transferred be \(a\). (Actually it is \(a.j.l\) but to avoid confusion with our convention we will call it \(a\)). We then have

\[(4.5.17) \quad I' - I = a\]

and

\[(4.5.18) \quad J^m = a.j.l - (a - J).\]

Substituting (4.5.17) and (4.5.18) in (4.5.16), we have

\[(4.5.19) \quad D = b[(1 + N*.i)(- a) + (1 + N*.j)J + (N*.j)a.j.l - N*.j(J - a + a.j.l)]
\]
\[= b[a(- 1 + N*.j - N*.i) + J] \geq 0.\]

At this point we have developed three types of job interchanges which will not increase the loss of a schedule. For the sake of clarity let us call machine \(j\) machine 1 and machine \(i\) machine 2 and summarize our results. If \(a.2.0 > a.1.1\), we may do one of the following without increasing the loss.

- **Case 1.** Interchange jobs 2.0 and 1.1. Job 2.0 then becomes job 1.1 and job 1.1 becomes job 2.0,

- **Case 2.1.** Interchange jobs 2.0 and 1.1. Job 2.0 then becomes job 1.1 and job 2.1 becomes job 2.0, or

- **Case 2.2.** Take the first job from machine 1 and put it first on machine 2. Job 2.0 remains job 2.0 but job 1.1 becomes job 1.0.

The conditions which determine which case to use have not been started since it will only be necessary to know that one of the cases can be used to make a job interchange.
Now consider the finite set \( Q \) of optimal schedules satisfying Theorem 3.3 on each machine. For any given schedule in this set let

\[ B_i \text{ be the set of job numbers on machine } i \text{ whose starting times are }< t.0 \text{ and let } A_i \text{ be the set of job numbers on machine } i \text{ whose starting times are } \geq t.0. \]

Let \( A = \bigcup_{i=1}^{K} A_i \) and \( B = \bigcup_{i=1}^{K} B_i \). We need to show that at least one schedule in \( Q \) has the property that

\[
(4.5.20) \quad \max_{i \in B} (a.i) \leq \min_{i \in A} (a.i)
\]

Suppose if possible that no schedule in \( Q \) satisfies \((4.5.20)\). Then every schedule in \( Q \) has at least one pair of job numbers \((i,j)\) such that job \( i \) starts before \( t.0 \) on some machine and job \( j \) starts on or after \( t.0 \) on some other machine and \( a.i > a.j \). Choose that schedule in \( Q \), say \( S' \), such that \( S' \) has the fewest number of pairs \((i,j)\) as defined above. Let the number of pairs \((i,j)\) in \( S' \) be \( N \). Based on this assumption, \( S' \) will be transformed using either Case 1, Case 2.1 or Case 2.2 without increasing the loss until it has only \( N - 1 \) pairs. This contradiction proves the theorem.

Let the set of pairs \((i,j)\) in \( S' \) be called \( P \). Among all pairs in \( P \) choose that pair \((i',j')\) such that \( j' = \min_{(i,j) \in P} (j) \), and for that \( j', i' = \max_{(i,j) \in P} (i) \). Now \( j' = \min A \) and \( i' = \max B \). Furthermore, \( j' \) must be the first job starting on or after \( t.0 \) on some machine and \( i' \) must be the last job starting before \( t.0 \) on some other machine. Since \( S' \) satisfies Theorem 3.3, they are not on the same machine.

Let \( j' \) be on machine 1 and \( i' \) be on machine 2. Job \( j' \) is then job 1.1 and job \( i' \) is job 2.0 and \( a.2.0 > a.1.1 \). We may now apply one of

\[ \text{Operation } U \text{ is the set operation union.} \]
the cases. If Case 1 is applied, \( j' \) goes into set \( B \) and \( i' \) goes into set \( A \) and since \( j' = \min A, i' = \max B \) and \( a_{11} \leq \ldots \leq a_{1N} \), we have reduced the number of pairs in \( P \) by at least one. If Case 2.1 is applied we may immediately follow it with Case 1 and the sum total of these two job interchanges will be to take \( j' \) from set \( A \) and put it in set \( B \). This will reduce the number of pairs in set \( P \) by at least one. If Case 2.2 is applied this will also take \( j' \) from set \( A \) and put it into set \( B \) which will also reduce the number of pairs in \( P \).

**Theorem 4.6.**—If the hypothesis of Theorem 4.5 holds, then there exists an optimal schedule which satisfies

\[
\begin{align*}
(4.6.1) \quad & N^{*1} \geq \ldots \geq N^{*K} \quad \text{and} \quad N^{*1} - N^{*K} \leq 1, \\
(4.6.2) \quad & \max_{(i|i \text{ starts before } t.0)} (a_{i1}) \leq \min_{(i|i \text{ starts on or after } t.0)} (a_{i1}) \\
(4.6.3) \quad & a_{i1} \leq \ldots \leq a_{iN}, \quad i = 1, \ldots, K,
\end{align*}
\]

and

\[
(4.6.4) \quad a_{11}^{*q} \leq \ldots \leq a_{h1}^{*q}, \quad 1 \leq q \leq N^{*1} \quad (h \text{ is the largest number such that machine } h \text{ has a } h^{*q}\text{th job})
\]

where \( a_{i1} \) is the operation time for the \( j \)th job on machine \( i \) and as in the convention defined in Theorem 4.4, job \( i.1^{*q} \) is the \( q \)th job starting on or after \( t.0 \) on machine 1.

**Proof.**—By Theorem 4.5 there exists an optimal schedule satisfying conditions (4.6.2) and (4.6.3). Since all optimal schedules satisfy Theorem 4.4, we may number the machines so that condition (4.6.1) is satisfied.

Now consider any schedule satisfying (4.6.1), (4.6.2), and (4.6.3), but not (4.6.4)\(^1\). It will be shown that this schedule can be transformed

\^[1]For example, if \( K = 2, t.0 = 3 \) and the sequence of operation times for the jobs on machine 1 is 1, 2, 3, 5 and for the jobs on machine 2 is 3, 4, 4, we have such a schedule.
without increasing the loss until it satisfies (4.6.4). For this schedule there exists and we can choose a smallest position number \( j \) and for that \( j \) a smallest machine number, say \( x \), such that \( a.x.j > a.y.j \) for some \( y > x \). We may choose \( y \) such that \( a.y.j \leq a.h.j, x < h \leq K \).

Let job \( x.0 \) finish \( X \) units late on machine \( x \) and job \( y.0 \) finish \( Y \) units late on machine \( y \). Then the loss on machine \( x \) is

\[
L(x.0, \ldots, x.N*.x) = \sum_{h=0}^{N*.x} b(X + a.x.1 + \ldots + a.x.h)
\]

\[
= b[(1 + N*.x)X + (N*.x)a.x.1 + \ldots + (1 - j + N*.x)a.x.j + \ldots + a.x.N*.x],
\]

and the loss on machine \( y \) is

\[
L(y.0, \ldots, y.N*.y) = \sum_{h=0}^{N*.y} b(Y + a.y.1 + \ldots + a.y.h)
\]

\[
= b[(1 + N*.y)Y + (N*.y)a.y.1 + \ldots + (1 - j + N*.y)a.y.j + \ldots + a.y.N*.y].
\]

If jobs \( x.j \) and \( y.j \) are interchanged, the loss on machine \( x \) is

\[
L(x.0, \ldots, x.j-1, y.j, x.j+1, \ldots, x.N*.x)
\]

\[
= L(x.0, \ldots, x.N*.x) + b[(1 - j + N*.x)(a.y.j - a.x.j)]
\]

and the loss on machine \( y \) is

\[
L(y.0, \ldots, y.j-1, x.j, y.j+1, \ldots, y.N*.y)
\]

\[
= L(y.0, \ldots, y.N*.y) + b[(1 - j + N*.y)(a.x.j - a.y.j)].
\]

The loss for the original schedule less the loss for the revised schedule is

\[
D = -b[(1 - j + N*.x)(a.y.j - a.x.j) + (l - j + N*.y)(a.x.j - a.y.j)]
\]

\[
= b[(a.x.j - a.y.j)(N*.x - N*.y)] \geq 0.
\]
It is possible that after interchanging jobs \( x.j \) and \( y.j \) condition (4.6.3) may not hold. The exact argument used in Theorem 4.2 will show that we may satisfy (4.6.3) after the job interchange. By repeating this entire procedure we can transform the whole schedule so that (4.6.4) is satisfied.

**Theorem 4.7.**—If the hypothesis of Theorem 4.5 holds, then there exists an optimal schedule in the set defined by Theorem 4.6 such that \( a.1.0 \leq \ldots \leq a.K.0 \) where \( i.0 \) is the last job on machine \( i \) whose starting time is \( < t.0 \).

**Proof.**—The class of schedules defined in Theorem 4.6 contains an optimal schedule. It will be shown that any optimal schedule satisfying Theorem 4.6 can be transformed without increasing the loss into a schedule satisfying Theorem 4.7. The proof will be in two parts. The first part will develop one type of job interchange which will not increase the loss and the second part will use this interchange to make the schedule transformation.

Suppose an optimal schedule in the set defined in Theorem 4.6 contains machines \( i \) and \( j \), \( i < j \), such that \( a.i.0 > a.j.0 \) and \( N*.i > N*.j \). Let job \( i.0 \) be \( I \) units late on machine \( i \) and job \( j.0 \) be \( J \) units late on machine \( j \). The loss on machine \( i \) is

\[
L(i.0, \ldots, i.N*.i) = \sum_{h=0}^{N*.i} b(I + a.i.1 + \ldots + a.i.h) = b[(1 + N*.i)I + (N*.i)a.i.1 + \ldots + a.i.N*.i],
\]

and the loss on machine \( j \) is
(4.7.2) \[ L(j_0, \ldots, j_{N^*j}) = \sum_{h=0}^{N^*_j} b(J + a_{j_0}l + \ldots + a_{jh}) \]

\[ = b[(1 + N^*_j)J + (N^*_j)a_{j_0}l + \ldots + a_{jN^*_j}] . \]

Suppose if possible \( a_{j_0} \geq a_{i_0} - I \). If jobs \( i_0 \) and \( j_0 \) are interchanged, let \( I' \) be the lateness of job \( j_0 \) on machine \( i \) and let \( J' \) be the lateness of \( i_0 \) on machine \( j \). Now \( J' > J \) and \( I' < I \) and since \( a_{j_0} \geq a_{i_0} - I, I' \geq 0 \). The loss equations for machines \( i \) and \( j \) for this revised schedule are the same as (4.7.1) and (4.7.2) except \( J' \) replaces \( J \) and \( I' \) replaces \( I \).

The loss for the original schedule less the loss for the revised schedule is

(4.7.3) \[ D = b[(1 + N^*_i)(I - I') + (1 + N^*_j)(J - J')] . \]

Noting that

(4.7.4) \[ I - I' = J' - J = a_{i_0} - a_{j_0} \]

and substituting (4.7.4) into (4.7.3), we have

(4.7.5) \[ D = b(a_{i_0} - a_{j_0})(N^*_i - N^*_j) > 0 . \]

This case cannot occur since the original schedule was optimal.

Therefore, it is the case that \( a_{j_0} < a_{i_0} - I \). If jobs \( i_0 \) and \( j_0 \) are interchanged the situation is the same as before except \( I' < 0 \). The loss equation for machine \( j \) is the same as (4.7.2) except \( J' \) replaces \( J \) but the loss on machine \( i \) is

(4.7.6) \[ L(i_1, \ldots, i_{N^*_i}) = \sum_{h=1}^{N^*_i} b(I' + a_{i_1}l + \ldots + a_{ih}) \]

\[ = b[N^*_i(I' + a_{i_1}) + \ldots + a_{iN^*_i}] . \]

The loss for the original schedule less the loss for the revised schedule is
\[(4.7.7) \quad D = b[(1 + N*.1)I - (N*.1)I' + (1 + N*.j)(J - J')] \]

Equation (4.7.4) still holds and noting that \(N*.1 = 1 + N*.j\), we have
\[(4.7.8) \quad D \geq b[(1 + N*.j)(I - I') + (1 + N*.j)(J - J') + 1] \geq 0.\]

To finish this case, it is possible that after changing jobs \(i.0\) and \(j.0\) condition (4.6.3) of Theorem 4.6 may not hold. By condition (4.6.2) of Theorem 4.6 and by the fact that \(a.i.0 > a.j.0\), condition (4.6.3) must hold on machine \(j\). By Theorem 3.3 we may interchange jobs \(j.0\) and the jobs preceding it on machine \(i\) to satisfy condition (4.6.3) without increasing the loss. (Note that after the interchange job \(i.1\) becomes job \(i.0\).)

Using this interchange an optimal schedule satisfying Theorem 4.7 will be constructed. Consider any optimal schedule in the set defined by Theorem 4.6. We begin by renumbering the machines so that if \(i < j\) and \(N*.i = N*.j\), then \(a.i.0 \leq a.j.0\). It is now possible that condition (4.6.4) of Theorem 4.6 is violated. If this is the case, then apply the procedure in the proof of Theorem 4.6 until condition (4.6.4) is satisfied. This transformation will not increase the loss for the schedule.

Now consider machine 1. If \(a.1.0 \leq a.i.0\), \(i = 2, \ldots, K\), then the theorem is satisfied for machine 1 and we have permanently established machine 1. If \(a.1.0 > a.i.0\) for some \(i > 1\), then there exists a \(j' > 1\) such that \(a.j'.0 \leq a.i.0\) \(i = 1, \ldots, K\). We know that \(N*.i > N*.j' \geq 0\) since the machines were numbered so that \(a.i.0 \leq a.j.0\) if \(N*.i = N*.j\) and \(i < j\).

By Theorem 4.6 we know that \(a.i.1 \leq a.i.j\), \(i = 1, \ldots, K; j = 1, \ldots, N*.i\), and that \(a.1.1 \geq a.i.0\), \(i = 1, \ldots, K\). Now interchange
jobs 1.0 and j'.0. Job 1.1 becomes job 1.0 and N*.1 is reduced by one unit from which it follows N*.1 ≤ N*.1, i = 2, ..., K. Renumber machine 1 to be machine K and let machine number i = i - 1, i = 2, ..., K. The schedule is still optimal and satisfies conditions (4.6.1), (4.6.2), and (4.6.3) of Theorem 4.6. Now apply the procedure in the proof of Theorem 4.6 until condition (4.6.4) is satisfied. This will not increase the loss and machine K is temporarily established.

We now have an optimal schedule satisfying Theorem 4.6 and we have established either machine 1 or machine K. If machine 1 was established in the first step then we consider machine 2. From this second step either machine 2 or machine K will be established. If we established machine K in the first step then we consider machine 1. (Machine 1 was machine 2 in the first step). From this second step we will establish either machine 1 or machine K. If machine K is established, then the previous machine K becomes machine K - 1. We can repeat this process a total of K - 1 steps until we obtain a complete ordering of the machines and the theorem is satisfied.

An application of this procedure is shown in Figure 12.

![Figure 12](attachment:image.png)

Although the original schedule is not optimal, it does show how the constructive proof is applied. In this example t.0 = \( \frac{41}{2} \) and the entries are the operation times for the seven jobs. Step one is the application
of the interchange developed in Theorem 4.7 to jobs 1.0 and 2.0. Step two is the renumbering of machine 1 to be machine 2 and reducing the number of machine 2 to 1. Step three is the application of Theorem 4.6. The schedule now satisfies Theorem 4.7.

Theorem 4.8.—Let there be K identical machines, a.1 \leq \ldots \leq a.H, f.i(x) = bx, b > 0, and d.1 = \ldots = d.H = t.0 > 0. Then there exists an optimal schedule such that

(4.8.1) a.1.0 \leq \ldots \leq a.K.0,

(4.8.2) The jobs whose starting times are \leq t.0 satisfy Theorem 3.3 on each individual machine,

(4.8.3) The jobs whose starting times are \leq t.0 have job numbers 1, 2, \ldots, M,

and

(4.8.4) The jobs whose starting times are \geq t.0 are placed according to the formula p.q = (q - 1)K + p + M, p = 1, \ldots, K; q = 1, \ldots, N^*.p, where N^*.p is the largest integer such that p.N^*.p \leq H.1

Proof.—Since there exists an optimal schedule in the set defined by Theorem 4.7 and all schedules in this set satisfy conditions (4.8.1), (4.8.2), and (4.8.3), it will be sufficient to show that any schedule in this set can be transformed without increasing the loss so that it still satisfies Theorem 4.7 and in addition satisfies condition (4.8.4).

Consider any schedule, S, in the set defined by Theorem 4.7 which does not satisfy condition 4.8.4. Then there exists a smallest position number, say j', and for that position number a smallest machine number, say i', such that i'.j' > (j' - 1)K + i' + M = w. Then job

---

1According to our convention p.q is the job number of the qth job on machine p whose starting time is \geq t.0.
number w must be in some other place in the schedule. Let job number
w be on the i"th machine in the j"th position. For the exact reasons
given in Theorem 4.3, a.i' .j' ≥ a.w ≥ a.i".j" and j" > j'. For this
schedule the loss on machine i' is

\[ L(i'.0, \ldots, i'.N*.i') = \sum_{h=0}^{N*.i'} b(I' + a.i'.l + \ldots + a.i'.h) \]

where job i'.0 is I' units late. The loss on machine i" is

\[ L(i".0, \ldots, i".N*.i") = \sum_{h=0}^{N*.i"} b(I" + a.i".l + \ldots + a.i".h) \]

where job i".0 is I" units late.

If jobs i'.j' and i".j" are interchanged, the loss on machine i' is

\[ L(i'.0, \ldots, i'.j'-1, i".j", i'.j'+1, \ldots, i'.N*.i') = L(i'.0, \ldots, i'.N*.i') + b[(1 - j' + N*.i')(a.i".j" - a.i'.j')] \]

The loss on machine i" is

\[ L(i".0, \ldots, i".j"-1, i'.j', i".j"+1, \ldots, i".N*.i") = L(i".0, \ldots, i".N*.i") + b[(1 - j" + N*.i")(a.i'.j' - a.i".j")] \]

The loss for the original schedule less the loss for the revised
 schedule is

\[ D = b[a.i'.j' - a.i".j"][(N*.i' - N*.i") + (j" - j')] ≥ 0. \]

The revised schedule is still optimal but to repeat the interchange
process we need to know that the revised schedule satisfies Theorem 4.7. It may violate conditions (4.6.3) and (4.6.4)\(^1\) of Theorem 4.6. The exact argument used in Theorem 4.3 can be used to show that these conditions can be satisfied without increasing the loss. At this point the entire procedure can be repeated until schedule \(S\) is transformed into a schedule which is in the set defined by Theorem 4.7 and satisfies condition (4.8.4).

It might appear that we would want to maximize \(M\) or even that Theorem 4.3 might hold, but the following example shows this not to be the case. Let \(t.0 = 3\), \(f.1(x) = x\) and \(a.1 = 1\), \(i = 1, \ldots, 5\). The best schedules for two machines are \(S: 1, 2, 4; 3, 5\) and \(S: 3, 4; 1, 2, 5\). For these schedules \(M = 3\) and the loss is nine. Theorem 4.3 maximizes \(M\) but gives a loss of ten. This schedule would be odd numbered jobs on machine 1 and even numbered jobs on machine 2.

Although Theorem 4.8 does not tell us how to get an optimal schedule, it considerably reduces the number of different schedules that one would have to evaluate. Once we know which jobs are assigned to each machine we could use Theorem 3.3 to tell us how to arrange them to get an optimal schedule. However, the number of ways we can assign \(H\) jobs to \(K\) machines, such that each machine has a job whose completion time is \(\geq t.0\)\(^2\), may be very large.

For \(H\) jobs, \(K\) machines, and a given \(t.0\), the actual number of such assignments may be quite difficult to compute especially since each

\(^1\)Note, these are the same conditions referred to in Theorem 4.3
\(^2\)If this condition does not hold for all optimal schedules, then the scheduling problem is practically trivial.
machine must have a job whose completion time is $\geq t_0$. If we disregard the condition that each machine must have a job whose completion time is $\geq t_0$ and replace it with the condition that each machine must have at least one job, then for simple cases we can fairly easily compute the number of different job assignments and be able to make an estimate of the number of different schedules we would have to evaluate using Theorem 3.3. For example, if $H = 10$ and $K = 2$, the number of assignments is $\binom{10}{1} + \binom{10}{2} + \binom{10}{3} + \binom{10}{4} + \frac{1}{2} \binom{10}{5} = 511$.

If we assume that each machine must have a job whose completion time exceeds $t_0$, then there exists a maximum value of $M$, say $\bar{M}$, and a minimum value of $M$, say $\underline{M}$. For each $M$, $\underline{M} \leq M \leq \bar{M}$, we can compute the number of ways we can assign the $M$ jobs and for each assignment evaluate the schedule using Theorem 4.8. If, as before, we only restrict the assignments to those such that each machine has at least one job, then again for simple cases we can fairly easily compute the number of different assignments. In the previous example problem if $\underline{M} = 4$ and $\bar{M} = 6$, the number of schedules we will have to evaluate is for $M = 4$, $\binom{4}{1} + \frac{1}{2} \binom{4}{2} = 7$; for $M = 5$, $\binom{5}{1} + \binom{5}{2} = 15$; and for $M = 6$, $\binom{6}{1} + \binom{6}{2} + \frac{1}{2} \binom{6}{3} = 31$. The total number of schedules is 53 which is considerably less than the 511 schedules that would have to be evaluated using Theorem 3.3.
CHAPTER V

SELECTION PROBLEM

Suppose a manager must select a field team from among his personnel and that the team must be able to perform a variety of tasks. The manager may find that some of his people can perform some of the tasks individually and that groups of his people can perform some of the tasks collectively but not individually.

One criterion the manager might use in selecting the personnel for the team would be the size of the team. That is, he might want to select the fewest number of people who individually or in groups could perform all the tasks.

The same problem might arise for the inventory manager of repair parts. Each repair item carried in inventory might be used in various pieces of equipment and many of the parts might be partial substitutes for other repair parts. He might want to stock the fewest number of different items that would be able to repair all of the equipment.

In a similar way this problem could arise for the manager who wants to start a job shop and advertise that his shop can perform a given list of jobs. He might want to find the fewest number of machines required to perform all the jobs or he might want to make the smallest capital outlay for a group of machines that could perform all the jobs. These are examples of a class of problems which will be called the selection problem.
More explicitly the problem may be stated as follows. We are given a set \( J = (j_1, \ldots, j_K) \) of jobs to perform and a set \( M = (m_1, \ldots, m_N) \) of resources to perform the jobs. We are also given the information to determine which resources or groups of resources can perform each job. The problem is to select the smallest subset of \( M \) which can perform all the jobs in \( J \).

Let the resources \( m_i \) be considered as Boolean variables. If \( m_i = 1 \), then \( m_i \) is selected and if \( m_i = 0 \), then \( m_i \) is not selected. Let us also consider the jobs \( j_i \) as being Boolean valued. If \( j_i = 1 \), then the job can be performed and if \( j_i = 0 \), then the job can not be performed. Several definitions will follow which will be used to define the problem more clearly.

**Definition 5.1.**—\( P = m_{i.1} \cdot m_{i.2} \cdot \ldots \cdot m_{i.h} \) denotes a Boolean product of resources. By the symbol \( m_i \in P \) or the statement \( m_i \) is contained in \( P \) we will mean that \( m_i \) is a variable in the product \( P \).

**Definition 5.2.**—\( P \) is said to "perform" job \( j_i \) if when the algebraic value of \( P \) is one, then the value of \( j_i \) is one. \( P = 1 \) when every \( m_i \) contained in \( P \) equals one and \( j_i = 1 \) when at least enough resources are selected to do the job.

In view of Definition 5.2 we can consider the information about each job \( j_i \) to be a set \( P(j_i) = \{P | P \text{ performs } j_i \} \). The problem is then to find a product \( P^* \) such that \( P^* \) performs \( j_i \) for all \( i \) and \( P^* \) contains the fewest number of resources. \( P^* \) may not be unique.

**Definition 5.3.**—\( P' \) is a multiple of \( P^* \) if there exists a \( P, P \neq 0, 1 \), such that \( P' = PP^* \).
Definition 5.4.—A "prime performer"\(^1\) of \(j.i\) is a performer of \(j.i\) which is not a multiple of another performer of \(j.i\).

In view of Definition 5.4 we may state the problem in an equivalent but different manner. Let \(P'(j.i) = \{P|P\text{ is a prime performer of } j.i\}\). The problem is then to find a \(P^*\) such that \(P^*\) is a multiple or identical to at least one \(P \in P'(j.i)\) for all \(i\) and \(P^*\) contains the fewest number of resources.

To enumerate enough combinations of resources to determine a \(P^*\) might require

\[
\sum_{i=1}^{N} \binom{N}{i} = 2^{N-1}
\]
calculations which for large \(N\) would be costly. Furthermore, one may well be interested in finding all products \(P\) such that \(P\) can perform all the jobs and \(P\) is not a multiple of another product which can perform all the jobs. Obviously this group of products contains all \(P^*\) and if a cost is associated with each resource, this group contains all minimum cost products which can perform all the jobs. From this discussion come the following definitions.

Definition 5.5.—A product \(P\) (the resources contained in \(P\)) is a solution for \(J\) if \(P \in P(j.i)\) for all \(i\).

Definition 5.6.—\(P\) is an admissible solution if \(P\) is a solution and \(P\) is not a multiple of another solution. The set \(A\) will be the set of all admissible solutions.

The rest of this chapter will be devoted to finding the set \(A\).

Job \(j.i\) equals one whenever we make a substitution of zeros and ones for

\(^{1}\)In the literature on symbolic logic and in other areas a widely used term is prime implicant.
the m.i, i = 1, ..., N, such that the m.i which equal one can perform job j.i. In this way job j.i defines a Boolean function which may be written algebraically or in "truth table" form. For example, if job j.i can be performed by m.1 alone or m.2 and m.3 together, the truth table for j.i is given in Table 1 and two of the many algebraic forms for j.i are:

(5.1) \[ j.i = \overline{m}.3 \cdot \overline{m}.2 \cdot m.1 + \overline{m}.3 \cdot m.2 \cdot m.1 + m.3 \cdot \overline{m}.2 \cdot m.1 + m.3 \cdot m.2 \cdot \overline{m}.1 + m.3 \cdot m.2 \cdot m.1 \]

and

(5.2) \[ j.i = m.1 + m.2 \cdot m.3. \]

| TABLE 1 |
| TRUTH TABLE |
| m.3 | m.2 | m.1 | j.i |
| 0   | 0   | 0   | 0   |
| 0   | 0   | 1   | 1   |
| 0   | 1   | 0   | 0   |
| 0   | 1   | 1   | 1   |
| 1   | 0   | 0   | 0   |
| 1   | 0   | 1   | 1   |
| 1   | 1   | 0   | 1   |
| 1   | 1   | 1   | 1   |

Theorem 5.1.—Let \( J^*(m.1, \ldots, m.N) = J^* = \prod_{i=1}^{K} j.i \) be a Boolean function where the j.i are represented in any of their algebraic forms. Then the set of admissible solutions for J is equal to the set of prime performers of J*.

1 Operation bar is the complimentation operation
Proof. — Let \( P \in A \). When \( P = 1 \) (that is, for any substitutions of zeros and ones for the m.i, \( i = 1, \ldots, N \), at least all of the m.i contained in \( P \) equal one), \( J^* = 1 \) since by definition \( P \in P(j,i) \) for all \( i \) and therefore, \( j.i = 1 \) for all \( i \). Since \( P \) is an admissible solution, none of its m.i may be dropped and all the \( j.i \) still equal one; therefore, \( P \) is a prime performer of \( J^* \).

Now let \( P \) be a prime performer of \( J^* \). When \( P = 1 \), \( j.i = 1 \), \( i = 1, \ldots, K \), since \( J^* = \prod_{i=1}^{K} j.i = 1 \); therefore, \( P \in P(j.i) \) for all \( i \). Since \( P \) is a prime performer of \( J^* \), no m.i in \( P \) may be dropped and \( P \) still perform \( J^* \), that is make all the \( j.i = 1 \); therefore, \( P \) is an admissible solution.

Two theorems will be used to help us find the prime performers of \( J^* \). The first due to Nelson (11) tells us that if \( J^* = \prod_{i=1}^{K} j.i \) where \( j.i = \sum_{P \in P'(j.i)} P \) is expanded and all multiples are dropped, the remaining terms are exactly all the prime performers of \( J^* \).\(^1\)

The second due to Quine (13) and Samson and Mills (14) relies on two definitions valid for any Boolean function.

**Definition 5.7.**—Given two products of Boolean variables, a variable is said to be in opposition if it appears in one product barred (operation bar is the complimentation operation) and in the other unbarred.

**Definition 5.8.**—If two products have exactly one opposition, the yield is the product of the variables not in opposition.

\(^1\)This statement of the theorem is modified for our particular problem.
For example, if \( P.1 = x.l \cdot \bar{x}.2 \cdot x.3 \) and \( P.2 = x.l \cdot x.2 \cdot x.3 \cdot x.4 \), \( x.2 \) is the opposition variable and the yield is \( x.l \cdot x.3 \cdot x.4 \). This theorem then tells us to begin with \( J^* \) in any algebraic form and

1. Drop all multiples.
2. If you can find a yield, add it to \( J^* \) and repeat (1) and (2). If you cannot find a yield, then \( J^* \) is represented as the sum of exactly all its prime performers.

**Sample problem.**—Let \( J = (j.1, j.2, j.3) \) where \( j.1 = m.4 \cdot m.1 + m.4 \cdot m.2 + m.3 \), \( j.2 = m.1 \cdot m.2 + m.4 + m.5 \), and \( j.3 = m.1 \cdot m.4 + m.5 \). In truth table form \( J \) and \( J^* \) appear in Table 2.

Using Nelson's theorem, we have

\[
(5.3) \quad J^* = (m.4 \cdot m.1 + m.4 \cdot m.2 + m.3)(m.1 \cdot m.2 + m.4 + m.5)
\]

\[
= m.4 \cdot m.2 \cdot m.1 + m.4 \cdot m.1 + \ldots + m.5 \cdot m.3.
\]

There are eighteen terms in \( J^* \), but after dropping multiples, we have

\[
(5.4) \quad J^* = m.4 \cdot m.1 + m.5 \cdot m.4 \cdot m.2 + m.5 \cdot m.3
\]

and \( A = (m.4 \cdot m.1, m.5 \cdot m.4 \cdot m.2, m.5 \cdot m.3) \).

A solution can also be obtained using Quine's theorem. In the \( J^* \) column of Table 2 we find fifteen entries of one. For each entry we can write a truth table product and then \( J^* \) can be represented as the sum of the fifteen truth table products. In this way we obtain

\[
(5.5) \quad J^* = \bar{m}.5 \cdot m.4 \cdot \bar{m}.3 \cdot \bar{m}.2 \cdot m.1 + \bar{m}.5 \cdot m.4 \cdot \bar{m}.3 \cdot m.2 \cdot m.1 + \ldots + m.5 \cdot m.4 \cdot m.3 \cdot m.2 \cdot m.1.
\]

By finding yields, adding them to \( J^* \), dropping multiples, finding yields, adding them to \( J^* \), etc., we finally obtain

\[
(5.6) \quad J^* = m.5 \cdot m.3 + m.5 \cdot m.4 \cdot m.2 + m.4 \cdot m.1.
\]

As before, \( A = (m.5 \cdot m.3, m.5 \cdot m.4 \cdot m.2, m.4 \cdot m.1) \).
<table>
<thead>
<tr>
<th>m.5</th>
<th>m.4</th>
<th>m.3</th>
<th>m.2</th>
<th>m.1</th>
<th>j.1</th>
<th>j.2</th>
<th>j.3</th>
<th>j*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
For very large problems, it would probably be necessary to put them on a computer both for speed and accuracy. For example, if there are ten jobs and each set \( P'(j,i) \) has ten elements, then using Nelson's theorem we would get \( 10^{10} \) terms after expanding \( J^* \). For the case when each \( P \in P'(j,i) \) contains only one m.i, current research,\(^1\) although not directed toward this problem, in the Mathematics Department at The Ohio State University can be applied to considerably reduce the calculations required by Nelson's theorem. This is a very restricted case and more research should be done for our more general problem.

\(^1\)This information was obtained through the courtesy of Larry D. Nelson and will appear in his dissertation when it is finished.
CHAPTER VI
CONCLUSIONS

The theorems developed for the sequencing problem can certainly be used to solve problems in the real world if the required conditions for the theorems are satisfied. However, it is felt that the practical application of this work may not be its most important contribution. Rather, it is believed that the methodology developed in the proofs of the theorems and the basic information resulting from the theorems, may be the most important contributions in terms of future research.

For practical applications this dissertation develops a general solution method for scheduling H jobs on one machine. For any increasing loss function Theorems 3.2 and 3.3 show that scheduling the shortest jobs first yields an optimal schedule if the deadlines for all jobs are the same. If the deadlines are different, Theorems 3.5 and 3.6 may be used to develop partial schedules.

For scheduling on K machines in parallel with a linear loss function, Theorem 4.3 develops an optimal schedule if the deadlines are $\leq 0$ and if the deadlines are all the same but positive, Theorem 4.8 limits the problem to a subset of schedules which contains an optimal one.

From this work on sequencing it appears that combinatorial solutions for very complex situations will be difficult to obtain. However, it is felt that this is the direction in which future research
should be directed using not only lateness as a criterion but also various other criteria. Because of the complexity of the problems, simulation will probably play a large part in future research.

A partial list of some specific problems is given below.

1. Include the weighting of jobs in the lateness criterion.
2. Apply the lateness criterion to several stages of production.
3. Consider K machines not necessarily identical.
4. Consider combinations of criteria such as lateness and machine utilization.
5. Improve the general method of Chapter II.

The selection problem solution method for large scale problems is costly and further research, directed toward finding better methods to obtain the prime performers of \( J^* \), is needed. It also appears that some of the ideas used in the selection problem might have wider application and some thought in this direction might be profitable.


I, James Gordon Root, was born in Columbus, Ohio, May 16, 1934. I received my secondary-school education in the public schools of Upper Arlington, Ohio, and my undergraduate and graduate training at The Ohio State University, which granted me the Bachelor of Industrial Engineering degree in 1957 and the Master of Science degree in 1961. From October, 1959, to June, 1962, I was an instructor in the Industrial Engineering Department. In October, 1962, I was appointed Battelle Memorial Institute Fellow at The Ohio State University, where I specialized in the department of Industrial Engineering.