ON DETERMINING CAPACITY LEVELS AT MACHINE CENTERS IN A JOB SHOP MANUFACTURING SYSTEM

By

STANLEY G. JONES

Bachelor of Science
Oklahoma State University
Stillwater, Oklahoma
1962

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1963

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 1968

ON DETERMINING CAPACITY LEVELS AT MACHINE CENTERS IN A JOB SHOP MANUFACTURING SYSTEM

Thesis Approved:

Wilson J. Bentley
Thesis Adviser

James E. Shamblin

Ingel J. Torgerson

Robert L. Sandquist

N. Durham
Dean of the Graduate College

PREFACE

Any job shop operation has a wide variety of operating problems due to the probabilistic nature of the production process. Two problems of considerable interest are (1) determining what the capacity requirements should be initially at each machine center and (2) providing a method for adjusting these capacities at the right time and by the correct amount to permit smooth operation of the shop. The stochastic nature of a job shop operation makes it very difficult to develop sound decision-making criteria. For this reason capacity decisions are often made on the basis of intuition and past experience. This method of management works since job shops do in fact operate. However, it does not usually provide a consistent method of operation, nor does it free management to concentrate its attention on only those matters that require a change from the present course of action.

Contribution of Study

The contribution of this study is that it provides statistically sound decision-making criteria for setting and adjusting capacity levels at machine centers in a job shop operation. A capacity requirement algorithm, using

capacity confidence limits for each machine center, provides the basis for the decision-making criteria. This algorithm determines the desired level of capacity at each machine center and also indicates when capacity changes should be made and the magnitude of these changes. The confidence limits are based on sound statistical concepts; however, in some instances simplifying assumptions are made for the sake of practicality.

Several simulation runs were made on an IBM 7090 computer using the Job Shop Simulator developed by the Oklahoma State University Operations Research Group. The results of these simulation runs show that the capacity requirement algorithm can be an aid in job shop management, provided proper values are selected for the algorithm's parameters.

Idea for Study

The idea for this study evolved as a result of a research grant to the Industrial Engineering and Management School at Oklahoma State University from the Industrial Engineering Section at the Wichita Branch of The Boeing Company. The purpose of this research grant was to study methods of improving Boeing's job shop operation. The use of the capacity requirement algorithm to set and adjust capacity levels at machine centers is only one of the job shop improvement techniques that has resulted from work done under this research grant.

Acknowledgments

My greatest debt of thanks goes to Wilson J. Bentley, Professor and Head of the School of Industrial Engineering and Management at Oklahoma State University, for his many kindnesses throughout my undergraduate and graduate career. In his various capacities as my teacher, as chairman of my graduate committee for both my MS and Ph.D. degrees, as my supervisor when I worked for the Industrial Engineering and Management School, and most important, as my friend, he has provided a constant source of encouragement, guidance, and counsel which have enabled me to grow and mature as an individual as well as advance in an academic sense.

Professor James E. Shamblin, who served as my adviser, deserves special thanks for his guidance and many suggestions during the preparation of this thesis.

In addition, I want to thank the remaining members of my graduate committee: Professors Earl J. Ferguson, Robert L. Sandmeyer, Wolter J. Fabrycky, and Paul E. Torgersen for their sound advice and guidance in setting up my plan of study, their helpful criticisms and suggestions while working on this thesis, and their genuine interest and encouragement throughout my graduate program.

I also want to thank Professor Robert W. Gibson for his help in preparing the simulation runs and the Industrial Engineering Section at the Wichita Branch of The Boeing Company for the research grant out of which came

the idea for this study.

A special note of thanks is required for Colonel Lloyd L. Dunlap, Jr., Professor and Head of the Department of Systems Management at the Air Force Institute of Technology. His ready availability for counsel and encouragement provided the much needed boost that was required to complete this thesis.

In addition, I want to thank two members of Colonel Dunlap's staff, Mrs. Janet Wheeler and Mrs. Carolyn Skelton, for their secretarial help in compiling the draft copies of this thesis.

Another special note of thanks is given to Miss Velda D. Davis for her great helpfulness in editing and typing this manuscript as well as for handling many of the countless details that are necessary in completing a graduate program.

A final debt which can never be paid is that which I owe my family and friends for the support, encouragement, and understanding they gave me during the preparation of this thesis.

Stanley G. Jones

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In all the literature on job shop operations, there is a noticeable lack of information concerning the setting and adjusting of capacity levels at machine centers. This problem does exist as evidenced by discussions with job shop production managers. However, as a general rule, it has always been handled by the production manager, relying on his past experience and intuition. This thesis is an attempt to establish a statistical approach to decision making criteria for the setting and adjusting of capacity levels at machine centers in a job shop operation.

Objective of Study

The objective of this study is to provide a sound decision making criteria for (a) determining what capacity requirements should be initially at each machine center in the job shop and (b) adjusting these capacities at the right time and by the correct amount to permit smooth operation of the shop. A capacity requirement algorithm using confidence intervals for each machine center provides the basis for this decision making criteria. This algorithm is

used to determine the desired level of capacity at each machine center and also to indicate when capacity changes should be made and the magnitude of these changes. The confidence intervals are based on sound statistical concepts drawing from established theory in quality control; however, in some instances, simplifying assumptions are made for the sake of practicality.

Plan of Presentation

This thesis is divided into three main parts. Chapter II is concerned with introducing the nature of job shop operations and acquainting the reader with the problems related to this type of manufacturing system. It explains a job shop process and describes some of its operating characteristics. The concept of flow time relationships is set forth with some empirical verification. The importance of determining an appropriate criteria for evaluating the performance of a job shop is pointed out and several examples of typical criteria are listed. Research efforts in various job shop problems are discussed. A review of the literature indicates that most of the effort has been directed toward solving the scheduling-sequencing problem. In fact, this particular problem is oftentimes referred to as "The Job Shop Problem." Examples of various approaches to this problem are set forth including direct enumeration, linear programming, simulation under priority rules, and

urgency number sequencing. The capacity requirement problem in job shop operations, which is the primary subject of this thesis, is defined and methods of solution, including intuition and experience, linear programming, and queueing theory, are described.

The development of the capacity requirement algorithm is given in Chapter III. In this chapter, the concept of the algorithm is set forth, its statistical basis is established, and all the constants and variables required to describe the system are defined. In addition, the algorithm itself is stated explicitly.

Chapter IV consists of an analysis of the results of simulation studies designed to test the effectiveness of the capacity requirement algorithm. These tests were run on an IBM 7090 computer using the Job Shop Simulator developed by the Oklahoma State University Operations Research Group. Several tests were conducted using different criteria for setting and adjusting capacity requirements, as well as different values for parameters and constants. These tests are described and their results evaluated.

A summary and conclusions chapter is included which also contains suggested topics for future research in the area of determining capacity levels at machine centers in a job shop manufacturing system.

CHAPTER II

AN INTRODUCTION TO JOB SHOP PROBLEMS

Until the advent of electronic data processing equipment, the computational problems related to job shop manufacturing systems presented such an overwhelming obstacle that heuristic methods were the basis for most analytical work done in this area. However, the availability of high speed computers has opened this area for further study. From their very beginning, job shop operations have always received considerable interest, but not until the past two decades has there been a method of analysis which could readily handle the immense computational problems associated with them.

The Nature of Job Shop Operations

Many manufacturing companies are, or have as a component of their total operation, a "job shop." Job shops may differ in size and complexity of operation, but they do have certain characteristics that identify them as a class of industrial manufacturing systems.

Description of a Job Shop

A job shop manufacturing system is usually

characterized by the physical arrangement of its equipment and its operating modes. The equipment is typically general purpose and arranged in groups according to the type of work performed as contrasted with a flow shop where the machines are arranged to manufacture a specific product. These groups of similar type equipment are called machine centers and are used to process a variety of manufacturing orders. Each order in the system is either waiting to be released or is already located at some machine center. The routing for each order is established by a routing sheet which involves a finite number of machine centers. The completion of a given order involves completing the operations described on its routing sheet, each operation requiring the use of machine time at the specified machine center.

The combination of set-up and operation time is often higher in a job shop as a result of the general purpose machinery and the variety of individual orders. These orders are variable in quantity, as well as type, and also have a wide range in value. This variety in orders affects both the inventory and queueing aspects of the production system. Manufacturing is also more variable due to new tooling, complex instructions, and the large variety of work. Because of this wide variability in equipment, orders, processing time, etc., a job shop manufacturing system can be viewed as a random variable process.

Flow Time Relationships

One of the important parameters of a job shop system is flow time. In general, there are two types of flow times, t_j which is the order flow time at any machine center j and T_i which is the total flow time for any order i. These two flow times are, however, interrelated.

Due to the stochastic nature of a job shop operation, order flow time at any machine center j is uncertain and may be described as a random variable. The components of order flow time can be roughly classified as follows: move time, m_j; queue time, q_j; set-up time, s_j; and operation time, o_j. Because of the multitude of random influences, each of these components is a random variable. Thus, order flow time at machine center j is a random variable expressed as

$$t_j = m_j + q_j + s_j + o_j.$$

The distribution of t_j may be determined empirically by reviewing the records of orders that have been processed at machine center j. The form of the distribution for t_j may vary depending upon the operating procedures at the particular machine center; however, a mean, μ_j, and a variance, σ_j^2, can be calulated for t_j from past data. Another method of obtaining values for μ_j and σ_j^2, is to make estimates of the order flow time for orders that are currently at machine center j. These estimates can then be used to

calculate μ_j and σ_j^2. A third method is to select some combination of past data and current data; e.g., make an estimate based on 30 per cent past data and 70 per cent current data.

Total flow time for any order i can be related to t_j in the following manner:

$$T_i = \sum_{j=1}^{n} t_j$$

where n refers to the machine centers on the routing sheet of order i. This expression says in effect that the total flow time for order i is equal to the sum of the order flow times of the machine centers through which order i must pass. Since t_j is a random variable, it follows that T_i will also be a random variable.

The application of the Central Limit Theorem makes it possible to develop an approximate distribution for T_i. It can be shown that T_i is distributed normally as n increases to infinity. This statement is true regardless of the form of the distribution of t_j if the following conditions are satisfied:

(a) The flow times, t_j, at machine centers are independently distributed random variables.

(b) The third absolute moment of t_j about its mean, ρ^3, is finite for every j.

(c) If $\rho^3 = \sum_{j=1}^{n} \rho_j^3$, then $\lim_{n\to\infty} \frac{\rho}{\sigma_i^2} = 0$ where $\sigma_i^2 = \sum_{j=1}^{n} \sigma_j^2$.

(d) The expected influence of any single t_j on T_i is relatively insignificant [1].

In the case of identically and independently distributed t_j, it is sufficient to require that the second order moment be finite for the Central Limit Theorem to be applicable. It is reasonable to assume that all of the above conditions are met in a large job shop subject to a multitude of random occurrences. Thus, the hypothesis is made that T_i is distributed approximately normal with a mean

$$\mu_i = \sum_{j=1}^{n} \mu_j$$

and a variance

$$\sigma_i^2 = \sum_{j=1}^{n} \sigma_j^2 .$$

If n is small, then the T_i distribution begins to assume the form of the t_j distributions that are used to calculate T_i.

Verification of Flow Time Relationships

Simulation runs were made on the IBM 1620 and IBM 1410 at Oklahoma State University in order to verify the relationships between order flow time and total flow time [2]. Figures 1 through 6 show the form of empirically developed (Monte Carlo Analysis) distributions of total flow time.

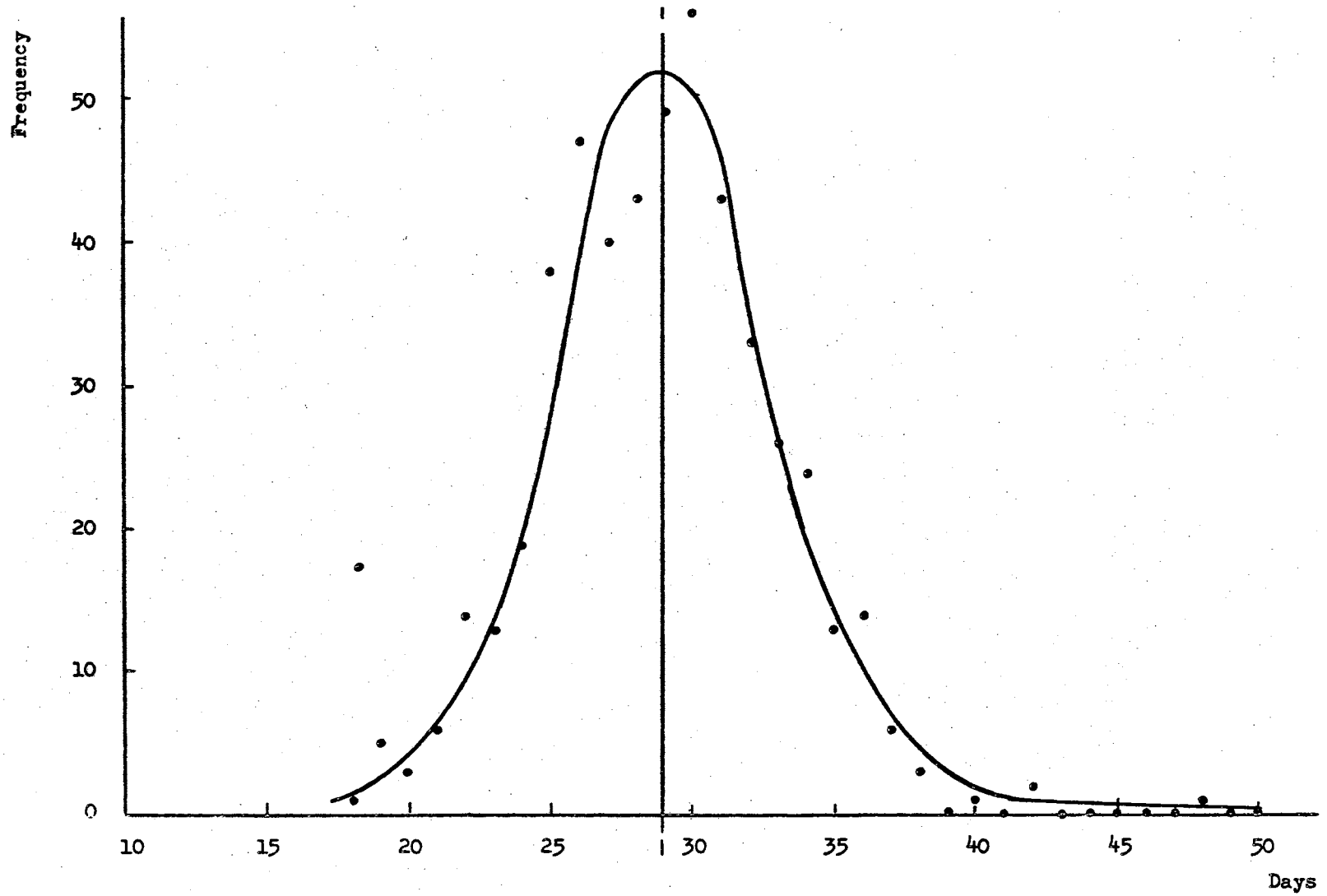

Figure 1. Normal Distribution - 10 Machine Centers

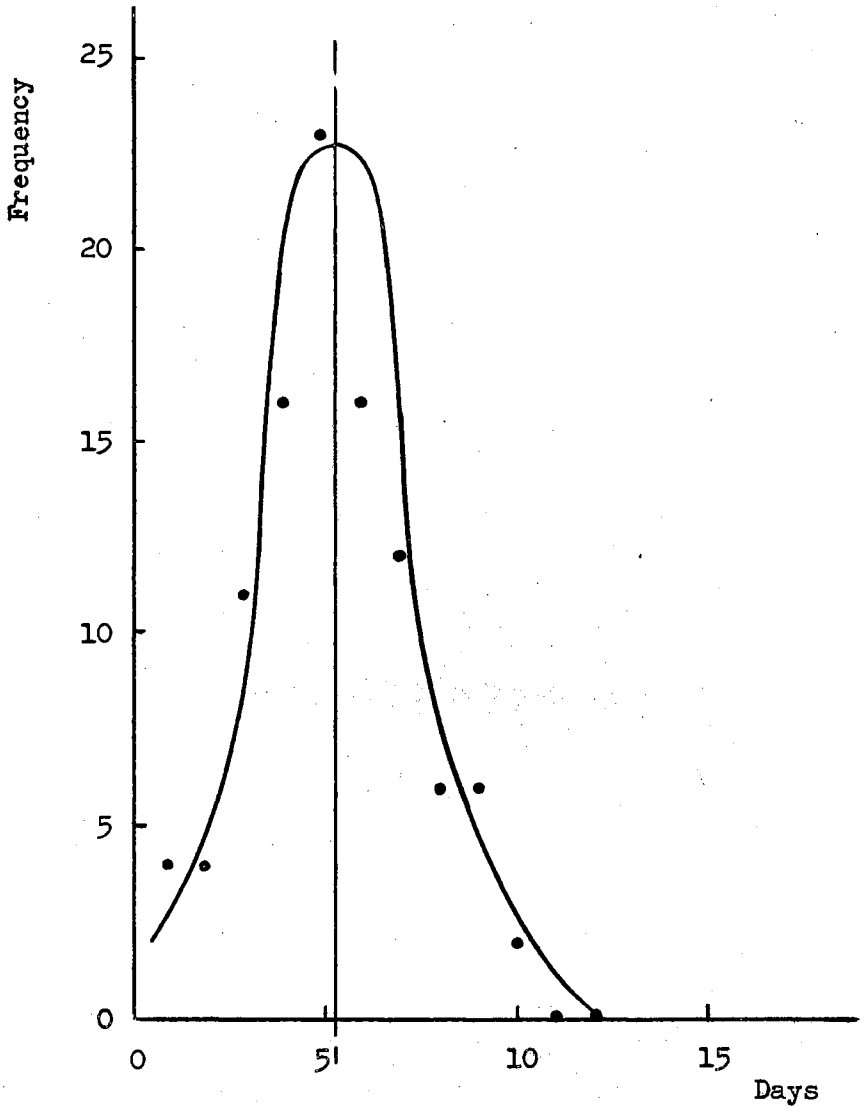

Figure 2. Normal Distribution - 2 Machine Centers

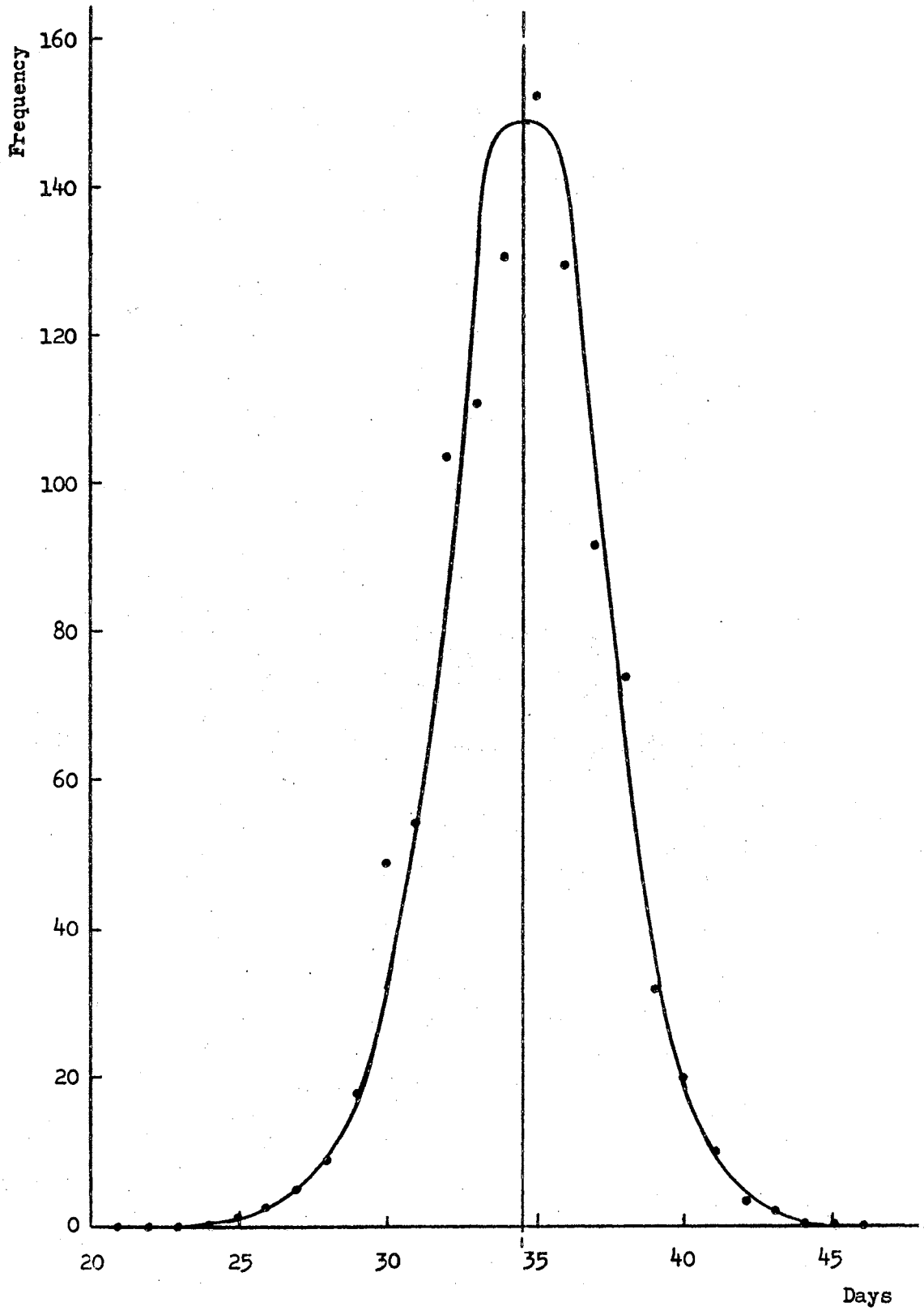

Figure 3. Rectangular Distribution - 10 Machine Centers

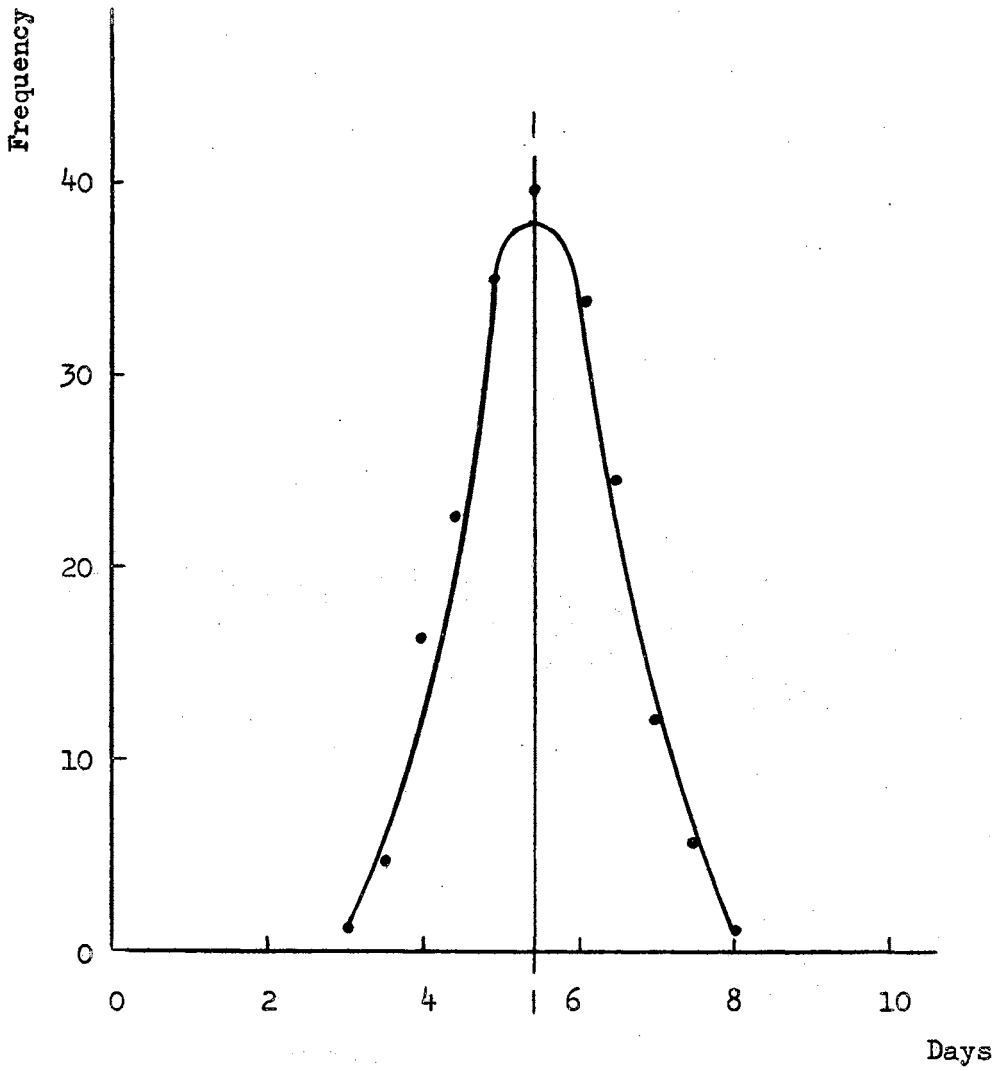

Figure 4. Rectangular Distribution - 2 Machine Centers

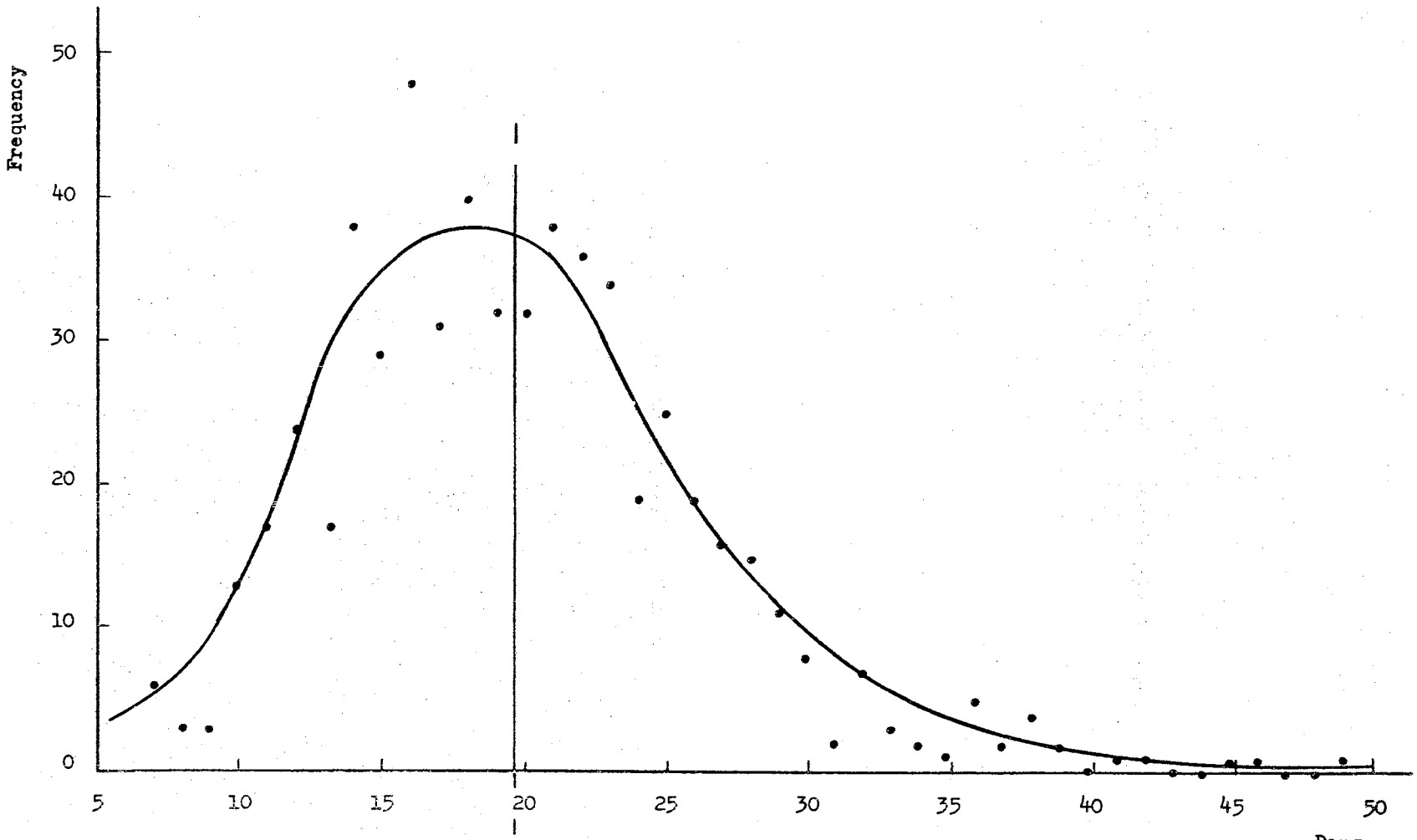

Figure 5. Exponential Distribution - 10 Machine Centers

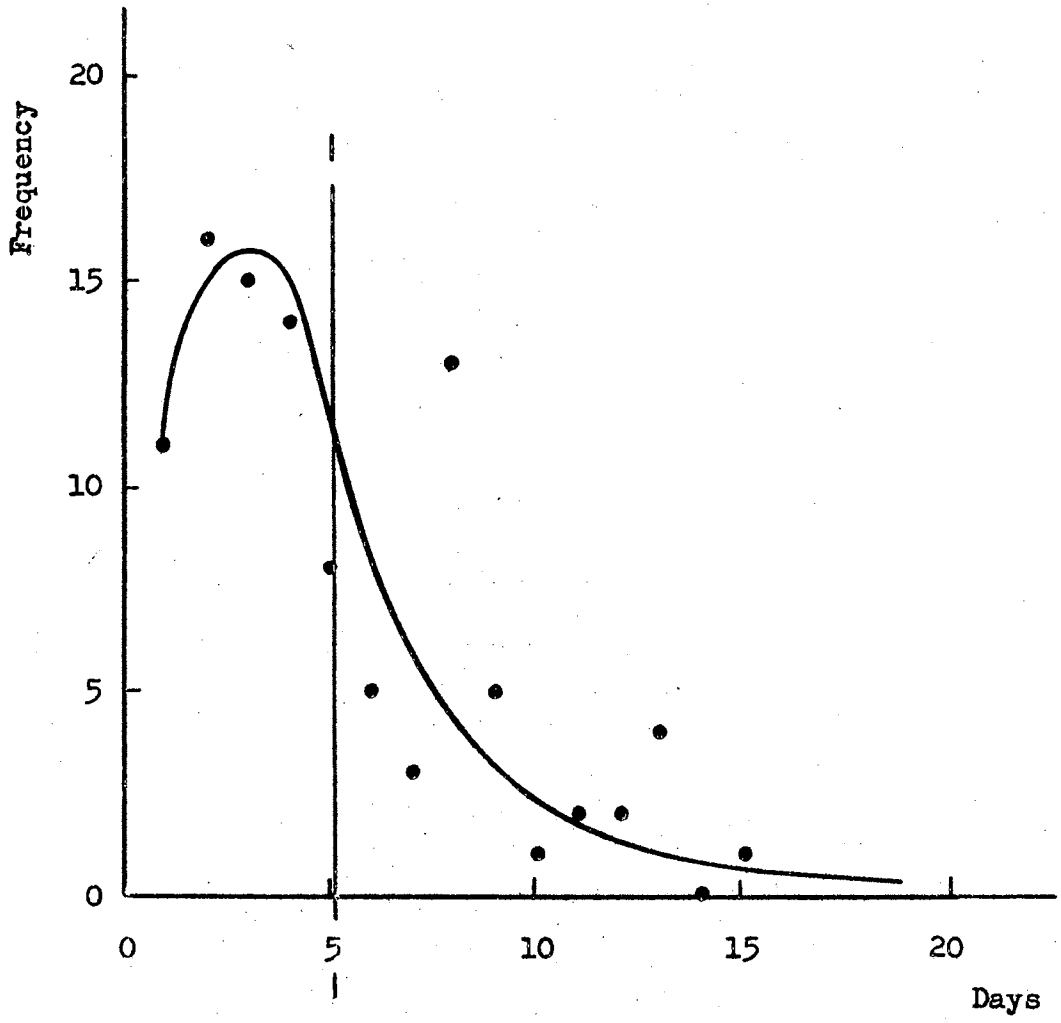

Figure 6. Exponential Distribution - 2 Machine Centers

The conditions under which these simulation runs were made are given in Table I.

TABLE I

CONDITIONS FOR TESTING DISTRIBUTION FORM

Figure	Number Machine Centers	Order Flow Time Distribution for Each Machine Center	Number of Observations
1	10	Normal	500
2	2	Normal	100
3	10	Rectangular	1000
4	2	Rectangular	100
5	10	Exponential	600
6	2	Exponential	100

The results of these simulation studies were statistically tested using a one-sided chi-square test at the 95 per cent confidence level. This was done in order to see how well the empirical distributions matched theoretical normal distributions. The following conclusions were reached. Figures 1 and 2 exhibited distributions that were well within the confidence limits; i.e., the hypothesis of normality was accepted. This result was expected due to

the fact that the flow times at the machine centers were distributed normally. The distribution given in Figure 3 also passed the chi-square test even though the flow times at the machine centers were distributed rectangularly. The distribution given in Figure 4 did not pass the chi-square test. There is a strong indication that additional observations and/or more machine centers would have made passing the test possible. The distributions in Figures 5 and 6 also failed to pass the chi-square test. The distribution in Figure 6 failed by a greater amount than the distribution in Figure 5. The primary reason for this deviation from normality was the skewedness of the order flow time distributions at the machine centers. However, there is strong indication that more machine centers and/or a larger number of observations would improve the convergence to normality.

These results fail to prove the normality assumption for the total flow time distribution when only a small number of machine centers is used. However, the results do indicate a convergence to normality as the number of machine centers increases. Greater convergence to normality is also exhibited when the order flow time at machine centers approaches a normal distribution.

The mean of each distribution given in Figures 1 through 6 was computed for comparison with the theoretical mean. The results are given in Table II.

TABLE II

COMPARISON OF MEANS

Distribution	Actual Mean	Theoretical Mean
Figure 1	28.90	29.00
Figure 2	5.26	5.00
Figure 3	34.50	34.50
Figure 4	5.44	5.50
Figure 5	19.77	20.00
Figure 6	5.07	5.00

From these results, it is concluded that the relationship $\mu_i = \sum_{j=1}^{n} \mu_j$ is valid. The slight deviations of the actual means from the theoretical means are due to sampling variation in the simulations.

The variance of each distribution given in Figures 1 through 6 was computed for comparison with the theoretical variance. The results are given in Table III.

Although the actual and theoretical variances do not agree as closely as the means, it is still concluded that the relationship $\sigma_i^2 = \sum_{j=1}^{n} \sigma_j^2$ is valid. As in the previous case, the differences in the actual and theoretical variances are due to sampling variation in the simulations. It should also be noted that the differences are greatest for the experiments with the fewest number of observations.

TABLE III

COMPARISON OF VARIANCES

Distribution	Actual Variance	Theoretical Variance
Figure 1	16.50	16.90
Figure 2	4.17	3.00
Figure 3	7.83	7.42
Figure 4	0.93	1.08
Figure 5	43.09	46.00
Figure 6	11.59	13.00

Performance Criteria

In running any industrial operation, management continually asks the question, "How well are we doing?" The answer to this question is usually expressed in terms of company profits, level of customer service, or some other criteria that is to be optimized; i.e., maximized or minimized. These broad objectives are then broken down into subobjectives for the operation of the business. In the case of a job shop system, some of the more common operating criteria are:

(a) Minimize the distribution of total flow times - the time from the introduction of the job to the shop to the completion of the last operation.

(b) Minimize the distribution of lateness of jobs - the length of time between actual completion of a job and the desired completion.

(c) Minimize the amount of work-in-process inventory in the shop.

(d) Minimize the amount of overtime in the shop.

(e) Maximize the utilization of shop facilities.

In actuality, total shop optimization is a function of all these subcriteria. However, the functional relationships between these conflicting objectives is extremely difficult, if not impossible, to find. Therefore, in actual practice, only the most critical subobjectives are considered and the system is operated in accordance with them; e.g., in a job shop operation, the most critical subobjectives might be minimizing job lateness without excessive overtime.

The Scheduling-Sequencing Problem

That aspect of a job shop operation which has received the most consideration is the scheduling-sequencing problem. This problem, simply defined, is to determine what jobs are to be assigned to which machine centers, as well as the sequence in which the work should be performed at each machine center, in order to optimize some criteria.

Criteria for performance were discussed earlier in this chapter.

The scheduling-sequencing problem can be approached in two general ways: (a) The exact approach for an optimum solution, and (b) the simulation approach for a near optimum solution.

Direct Enumeration

Exact optimum solutions to the scheduling-sequencing problem have been the object of much investigation by theoretical analysts. The most obvious exact approach to an optimum solution of the scheduling-sequencing problem is direct enumeration. This approach is simple in that all possible alternatives are considered and the best is selected; however, it becomes quite unmanageable due to the computational difficulties involved [3]. In general, there are N jobs and M machines. Each job has a given order of operations on some or all of the M machines with given processing times. There are $(N!)^M$ possible schedules. Of these, some are not feasible because they conflict with the prescribed routings. Of the feasible set of schedules, the problem is to select the schedule or schedules that optimize some desired quantity.

The computational immensity of just the active feasible schedules, which is a much smaller set than the set of all feasible schedules, is illustrated by the "6 x 6 problem" in Figure 7. This scheduling-sequencing problem

is referred to as a "6 x 6 problem" because six jobs are processed over one or more of six facilities.

Jobs	Facility					
1	3	1	2	4	6	5
2	2	3	5	6	1	4
3	3	4	6	1	2	5
4	2	1	3	4	5	6
5	3	2	5	6	1	4
6	2	4	6	1	5	3

Figure 7. Facility Order Matrix

The facility order matrix in Figure 7 is used to generate a series of problems referred to as "6 x 6*1", "6 x 6*2", ..., "6 x 6*6". The first of these problems is obtained by considering that each job is produced by one operation on the facility indicated by column 1 of the facility order matrix. Similarly, the second problem is obtained by considering that each job is produced by performing two operations on the facilities indicated in columns 1 and 2, etc. As might be expected, the number of active feasible schedules in each of these problems increases very rapidly. The actual number of active feasible

schedules for these problems is given in Table IV.

TABLE IV

NUMBER OF ACTIVE FEASIBLE SCHEDULES

Problem	Number of Active Feasible Schedules	Time to Solve (Min.)
6 x 6*1	36	0.00
6 x 6*2	290	0.09
6 x 6*3	914	0.48
6 x 6*4	7,546	4.82
6 x 6*5	84,802	70.18

Observe that, in the 6 x 6*5 case, a combinatorial problem is defined by 30 integers, each between 1 and 6. The resulting problem kept an IBM 704 computer busy for 70 minutes producing 84,802 active feasible schedules. In examining some of the schedules produced, it is estimated that there are approximately 100 inactive feasible schedules for every active feasibe schedule. Thus, the total number of feasible schedules for the 6 x 6*5 problem is approximately eight and one-half million. For obvious reasons, complete enumeration for the 6 x 6*6 problem was not attempted.

Linear Programming Approaches

Another attempt to obtain an exact optimum solution to the scheduling-sequencing problem can be formulated using the concepts of linear programming. Any linear programming solution is based on the fact that the problem can be stated in the form of a set of linear constraints (equality or inequality). In addition, a linear objective function employing the same variables as the constraints is required.

One such approach was set forth by E. H. Bowman [4]. In this approach, the restrictions characterizing a specific scheduling-sequencing problem are represented by two matrices. For illustrative purposes, specific jobs, machines, and time notations are used. Let the jobs be x, y, and z; the machines be A, B, C, D; and the time periods (small) run from 1, 2, 3, ..., T.

The required order of operation for job x is A, B, C, D; for job y is C, A, B, D; and for z is D, A. The facility order requirement of each job is given by the facility order matrix in Figure 8.

Job	Facility			
x	A	B	C	D
y	C	A	D	B
z	D	A		

Figure 8. Facility Order Matrix

The manufacturing times (set-up plus operation) required (in time period units) are given by the operation time matrix in Figure 9.

		Facility			
		A	B	C	D
Job	x	5	2	8	7
	y	4	3	8	5
	z	7	0	0	6

Figure 9. Operation Time Matrix

In the facility order matrix of Figure 8, a row corresponds to one of the N jobs and a column corresponds to a work station for the manufacture of the job at a specific machine location. For example, if job x must be processed on machines A, B, ..., M in that order, the i^{th} row entry at work station A will be defined as "machine A," at work station B as "machine B," ..., at work station M as "machine M." If a job is not to be processed on every machine, then the number of work stations is less than M.

For sake of simplicity, it is postulated that the processing specifications for any job are such that any

machine does not appear more than once, if at all, in row i of the facility order matrix. Some modification of the model would be needed to allow for multiple processing.

In the operation time matrix of Figure 9, the rows correspond to the jobs and the columns to the machines. The entry, t_{ik}, at the intersection of row i and column k represents the set-up plus operation time for job i on machine k. If the facility order matrix indicates that a particular job is not to be scheduled on a particular machine, then the corresponding element in the operation time matrix is assumed to be zero. It is further assumed that the time units are measured such that every t_{ik} is an integer.

The basic variables in the formulation are of the form $x_{A:1}$ meaning that job x requires machine operation A during time period 1. These variables take on the values zero or one in the formulation; i.e., the job is or is not worked on during this period. The form of the constraints is:

$$1 \geq x_{A:1}, x_{A:2}, \cdots, x_{A:T}, x_{B:1}, x_{B:2}, \cdots, y_{D:T},$$

$$\cdots, z_{D:T} \geq 0. \qquad (1)$$

It is necessary to include constraints assuring that the individual operations will be performed. For example, product x requires five time units of processing on machine A, two time units of processing on machine B, etc. The form of the constraints is:

$$
\begin{aligned}
x_{A:1} + x_{A:2} + \cdots + x_{A:T} &= 5 \\
x_{B:1} + x_{B:2} + \cdots + x_{B:T} &= 2 \\
\vdots \qquad \vdots \qquad & \qquad \vdots \\
y_{A:1} + y_{A:2} + \cdots + y_{A:T} &= 4 \\
\vdots \qquad \vdots \qquad & \qquad \vdots \\
z_{D:1} + z_{D:2} + \cdots + z_{D:T} &= 6.
\end{aligned} \tag{2}
$$

Two or more products may not be processed by the same machine at the same time; i.e., conflicting assignments are forbidden. The form of the constraints is:

$$
\begin{aligned}
x_{A:1} + y_{A:1} + z_{A:1} &\leq 1 \\
x_{A:2} + y_{A:2} + z_{A:2} &\leq 1 \\
\vdots \qquad \vdots \qquad & \qquad \vdots \\
x_{D:T} + y_{D:T} + z_{D:T} &\leq 1.
\end{aligned} \tag{3}
$$

Proper sequencing is the key part of this problem. No operation may be undertaken until the previous operation on the job in the specified sequence has been completed in a previous time period. For example, job x requires five time units on machine A before its operation on machine B can be started. This operation on machine B (two time units), in turn, must precede the operation on machine C.

The form of the constraints is:

$$
\begin{aligned}
5x_{B:j} &\leq \sum_{i=1}^{j-1} x_{A:i} \\
2x_{C:j} &\leq \sum_{i=1}^{j-1} x_{B:i} \\
&\vdots \\
6z_{A:j} &\leq \sum_{i=1}^{j-1} z_{D:i}
\end{aligned}
\qquad (4)
$$

for all $j = 1$ to $j = T$.

There is no guarantee in the above formulation that operation runs will not be interrupted, only that sequencing will be correct. If operation runs must not be broken (because of set-up costs, for example), then additional sets of constraints such as the following can be added:

$$
\begin{aligned}
5x_{A:i} - 5x_{A:i+1} + \sum_{j=i+2}^{T} x_{A:j} &\leq 5 \\
2x_{B:i} - 2x_{B:i+1} + \sum_{j=i+2}^{T} x_{B:j} &\leq 2 \\
&\vdots \\
6z_{D:i} - 6z_{D:i+1} + \sum_{j=i+2}^{T} z_{D:j} &\leq 6
\end{aligned}
\qquad (5)
$$

for all $i = 1$ to $i = T$.

These constraints do not allow a "one" variable to be followed by a "zero" variable and yet be followed by more "one" variables. Feasible scheduling is, therefore, not excluded. For instance, the assignment of product x to machine A in the time sequence

1	2	3	4	5	6	7	8	9	(time)
0	0	1	1	1	0	0	1	1	(assignment)

is excluded because

$$5x_{A:5} - 5x_{A:6} + \Sigma(x_{A:7} + x_{A:8} + x_{A:9}) = 5 - 0 + 2 = 7$$

which is not ≤ 5.

To obtain a solution to the linear programming problem, the variables of the form $x_{A:1}$ must have values associated with them (many such values may be zero). In a sense, the objective is to have the final operations on all products performed as early as possible. Prior operations, such as all those on machine C, will, of course, have preceded the final operations. The following objective function is suggested to be minimized:

$$\begin{aligned}\text{Objective function} = {} & 1(x_{D:23} + y_{B:23} + z_{A:23}) \\ & + 4(x_{D:24} + y_{B:24} + z_{A:24}) \\ & + 16(x_{D:25} + y_{B:25} + y_{A:25}) \\ & + 64(x_{D:26} + y_{B:26} + z_{A:26}) + \cdots \\ & + K_T(x_{D:T} + y_{B:T} + z_{A:T}) \end{aligned} \qquad (6)$$

where $K_T = 4K_{T-1}$. The rational of this objective function is that it makes operations (the last ones on each product) toward the end of the time periods costly. The number of time periods, chosen in advance of solution, may be equal to or less than the sum of all operation times (55 time units) but cannot be less than the sum of operation times required on the longest product (22 time units). The cost associated with any operation in a time period is a synthetic cost equal to the sum of all prior costs plus one. This exploding cost function thus forces operations toward the beginning for economic reasons. No later time period will be ultimately used than the minimum (optimal) as this one cost is larger than the sum of all prior costs. That is, given some feasible solution, the latest (and last) operation would be moved earlier by one time period. All other operations could be moved later by any number of time periods (excluding movement into or beyond the original last time period) and the exchange would be favorable. A problem where different specific costs can be assigned to uncompleted products beyond certain dates would, of course, not require the generation of synthetic costs.

Even the simple problem presented here for illustration has from 300 to 600 real variables depending on the number of time periods chosen. The number of constraints is substantially larger. Approaches to reducing the computational complexity of a linear programming solution include Gomory's [5] bounding procedures for choosing the

number of time periods, Dantzig's [6] method of choosing grosser units for time period length than the sensitivity of measurement available, Markowitz and Manne's [7] elimination of obvious redundancy in some of the constraints, and Andrus and Beker's [8] use of the on-off nature of all the variables.

Another linear programming approach to the scheduling-sequencing problem was developed by A. S. Manne [9]. In this approach, an attempt is made to reduce the large number of variables and constraints. The formulation of the problem is given below.

Let x_j be defined as an unknown integer value indicating the day that job j is to be started ($x_j = 0, 1, 2, \ldots, T$). Suppose that jobs j and k require a_j and a_k consecutive days, respectively. If these jobs are to be prevented from occupying the same machine at the same time, one of the two must precede the other by sufficient time in order for the first one to be completed before the second is started. This situation can be stated as either

$$x_j - x_k \geq a_k$$

or

$$x_k - x_j \geq a_j. \tag{7}$$

In order to convert this either-or condition into a linear inequality in integer unknowns, it is convenient to define a new integer-valued variable, y_{jk}, and to write the

following constraints:

$$0 \leq y_{jk} \leq 1 \tag{8}$$

$$(T + a_k)y_{jk} + (x_j - x_k) \geq a_k \tag{9}$$

$$(T + a_j)(1 - y_{jk}) + (x_k - x_j) \geq a_j. \tag{10}$$

Condition (8) insures that y_{jk} equals either zero or unity. It is already known that $|x_j - x_k| \leq T$. The effect of Conditions (9) and (10) may be summarized as follows:

$$\text{If } (x_j - x_k) \begin{Bmatrix} > 0 \\ = 0 \\ < 0 \end{Bmatrix}, \text{ then } y_{jk} = \begin{Bmatrix} 0,1 \\ 1 \\ 1 \end{Bmatrix} \text{ and } y_{jk} = \begin{Bmatrix} 0 \\ 0 \\ 0,1 \end{Bmatrix},$$

where the first set of values for y_{jk} is implied by Condition (9) and the second set by Condition (10).

Hence, if $(x_j - x_k) = 0$, there is no value that can be assigned to y_{jk} so that both (9) and (10) will be satisfied. If, on the other hand, $(x_j - x_k) \neq 0$, y_{jk} will be set at a value of either zero or unity depending upon which job is to precede the other. Conditions (9) and (10) then insure that the first job will be initiated in sufficient time to be completed before the beginning of the second one. Note that, with the classical form of linear programming, it would have been impossible to specify such an either-or condition as (7). This noninterference restriction leads directly to a nonconvex set of constraints upon the unknowns.

Once the noninterference stipulations have been established, the remainder of the formulation becomes virtually automatic. If job j is to precede job k, this means that job k is to be performed at least a_j days later than job j. The integer programming condition for this sequencing constraint is:

$$x_j + a_j \leq x_k. \tag{11a}$$

"Weak" precedence relations may be written in an analogous fashion. For example, in order to specify that both jobs i and j precede k, but that there are no precedence constraints affecting the performance of i and j, the following constraints would be written:

$$x_i + a_i \leq x_k$$

$$x_j + a_j \leq x_k. \tag{11b}$$

Still another possibility might be that there would be a delay of exactly θ_{jk} days between the performance of jobs j and k. Such a constraint would be indicated by:

$$x_j + a_j + \theta_{jk} = x_k. \tag{11c}$$

Specific delivery date requirements may be imposed on the system. For example, it may happen that the shop is committed to the delivery of an individual job no later than a specified date. If task j is the last task which

the shop is to perform upon the job and if the job is to be available on day d_j, this form of constraint may be written:

$$x_j + a_j \leq d_j. \tag{12}$$

In this linear programming formulation of the scheduling-sequencing problem, the criteria for optimality is the "make-span" or total calendar time as denoted by t. The problem now consists of minimizing t with respect to the nonnegative integers x_j and y_{jk} subject to Conditions (8) through (12) and also subject to the over-all delivery requirement:

$$x_j + a_j \leq t \qquad (j = 1, \ldots, n). \tag{13}$$

Excluding all of the slack variables and also t, the number of unknowns in this formulation is equal to the total number of the x_j plus the y_{jk}. If, then, there are n tasks and m possible conflicting pairs of machine assignments, the total number of unknowns would come to n + m. For example, with five machines and ten tasks to be performed on each machine, n = (10)(5) = 50 and m = $\frac{1}{2}$(5)(10)(10-1) = 225. The total number of integer-valued unknowns, x_j and y_{jk}, would come to 275. If an algorithm were available to handle "mixed" integer programming problems; i.e., problems in which some of the unknowns are constrained to take on integer values and others are permitted to be continuous, this scheduling model would fit very naturally

into the category of such a "mixed" problem. The y_{jk} unknowns are necessarily of a discrete nature; however, it might be more efficient, and possibly more realistic, to regard the start dates, x_j, as continuous variables.

In further work along these lines, one of the most important avenues to be explored is the possibility of reducing the number of unknowns, y_{jk}. Aside from the upper bound constraints, Condition (8), these unknowns are involved only in connection with the machine interference conditions, (9) and (10). Since many of these restrictions will inevitably be redundant in any particular numerical problem, it might be quite feasible to apply a computer code designed around Dantzig's [10] principle of "secondary constraints." In the traveling salesman problem, for example, one does not write down explicitly all conceivable loop constraints, but only those that have been violated during the course of previous iterations. By applying this same principle to the scheduling-sequencing problem, this suggestion may conceivably make it economical to obtain exact solutions to realistic examples.

Simulated Experimentation Under Priority Rules

The exact approaches (complete enumeration and linear programming) for solving the job shop scheduling-sequencing problem have met with limited success. The most prominent difficulty encountered by the exact algorithms is that the

computational difficulties tend to increase rapidly with the size of the problem.

Lacking a practical algorithm to solve for the exact optimum schedule for the processing of many jobs through a given set of machines, one must rely upon simulation techniques. Simulation, in the context of scheduling theory, generates and evaluates many schedules and chooses the "best" schedule; i.e., the minimum of some function of the schedule time or some other criteria for optimality (discussed earlier in this chapter).

Generally, simulation studies are used to evaluate the effect of some priority rule. That is, given a certain job shop situation and criteria for optimality, which priority rule results in the best performance?

A priority rule, simply stated, is a method for determining which job to work next. For example, the first-come first-served rule says to select that job which was first to arrive in the queue. A countless number of such rules can be formulated, depending upon the objectives of the operation. There are several interesting ways of classifying priority assignment rules. One can categorize procedures according to their information horizons. They can be segregated as strictly local procedures in which the priorities are entirely a function of characteristics of the particular job in question. These do not depend, in any way, upon the status of the shop or the presence or absence

of characteristics of other jobs. The following is a list of some priority rules that can be used [11]:

(1) Priority value assigned at random.

(2) Priority given in arrival order. The first arrival in a queue receives the highest priority.

(3) Priority value which is inversely related to the due-date of the job. The due-date being that date on which the job should be completed. The job with the earliest due-date has the highest priority.

(4) Priority which is inversely related to the remaining slack time. Slack time is the time remaining between the due-date and the remaining processing time. The job with the minimum slack time is given top priority.

(5) Priority which is inversely related to the processing time on the next operation. Maximum priority is given to the job with the shortest operation time on the machine in question.

(6) Priority which is directly related to the processing time on the current operation. Maximum priority is given to the job with the longest operation time on the machine in question.

(7) Priority which is inversely related to the number of remaining operations. Maximum priority is given to the job with the fewest remaining operations.

(8) Priority which is directly related to the number of remaining operations. Maximum priority is given to the job with the most remaining operations.

(9) Priority which is inversely related to the total remaining processing time. Maximum priority is given to the job for which the sum of the processing times for all the remaining operations is a minimum.

(10) Priority which is directly related to the total remaining processing time. Maximum priority is given to the job for which the sum of the processing time for all the remaining operations is a maximum.

(11) Priority which depends upon the dollar value of the job. Jobs are divided into two classes, a high-value class and a low-value class. All high-value jobs are assigned greater priorities than all low-value jobs. Within the class, priority is assigned in arrival order.

(12) Priority which is directly related to the dollar value of the job.

(13) Priority which is related to the subsequent move. Maximum priority is given to that job which, on leaving this machine center, will go to the next machine center which has the shortest (in the sense of least processing time) critical queue. If no queue is considered critical, the selection is by arrival order. A queue is considered critical when it has less than a specified number of time units of processing time waiting.

Recent work by Conway [12] has evaluated the effect of a number of priority rules on a typical job shop operation.

In preparing for simulated experimentation with a production problem, the first step is to construct a conceptual model which represents the manufacturing system in mathematical and logical terms. This model may be very simple, omitting many details of the actual situation and idealizing the rest, or it may be quite involved. Since the scheduling-sequencing problem is concerned with detailed operations within a plant, the appropriate models are generally quite complex.

The second step is to program a computer to simulate the operation of the manufacturing system by operating the model. The complexity of realistic scheduling-sequencing problems requires the use of high speed electronic computers

in performing the simulation. However, for illustrative purposes, a small, artificial problem developed by Rowe and Jackson [13] will be presented.

Consider the following situation. Three jobs, J-1, J-2, and J-3, are to be processed by a plant having three machines, M-1, M-2, and M-3. The sequence of operations on each job is completely determined and it is assumed that the processing time required for each operation is known. It has also been decided that each job will be processed as a single lot; i.e., lot splitting will not be allowed. This data along with data related to the scheduling methods to be considered is given in Figure 10.

Each job is available for its first operation at time zero and for each subsequent operation at the moment the preceding operation is completed. A job can be assigned to the machine designated for a given operation at the time it becomes available for the operation or at any later time. It is required that a machine work whenever a job is available for processing; however, a machine can work on only one job at a time. The time needed for transportation is assumed to be negligible and the possibility of machine breakdown is excluded from the model.

These assumptions have been listed to indicate explicitly how the conditions of a problem should be defined before a conceptual model can be used. There is no limit to the complexities that can be put into such a model, but

no detail can be expected to appear unless it is explicitly stated. In addition, care must be taken that unnoticed limitations are not implicitly included in the model.

In this example, the problem is to determine which of two priority rules will result in the completion of all three jobs by the earlier date. The assumptions listed above imply that a priority rule will be completely determined if a method is given for deciding which of a number of jobs will be assigned to a machine next whenever a conflict arises. Priority Rule 1 requires that the job be assigned for which the total remaining processing time is greatest. Priority Rule 2 requires the assignment of the job for which the remaining processing time, excluding that for the operation under consideration, is greatest.

The model is based upon a Gantt chart; i.e., a chart with bars that can be filled in to indicate the scheduled activity for each machine at each moment in time. The restrictions on filling out this chart are:

(a) Each operation must be assigned an interval on the bar representing the machine called for by the routing. The length of this interval is proportional to the processing time.

(b) The intervals assigned to a single job must not overlap and must be performed in the required sequence.

(c) The intervals assigned to different operations on the same machine must not overlap.

The model actually consists of the chart along with the stated restrictions on the way it can be used.

The computational arrangement for this example is now described. Each job is represented by a job-card on which the job's relevant data is recorded. These cards list the required operations in the required sequence, the designated machines, the processing times, and the priority numbers to be used for each of the two methods of scheduling. The data for these cards is given in Figure 10.

Job	Operation Numbers	Machines	Required Times (days)	Priority Numbers Rule 1	Rule 2
J-1	J-1-1	M-1	3	10	7
	J-1-2	M-2	5	7	2
	J-1-3	M-3	2	2	0
J-2	J-2-1	M-1	6	11	5
	J-2-2	M-2	2	5	3
	J-2-3	M-3	3	3	0
J-3	J-3-1	M-2	5	9	4
	J-3-2	M-3	4	4	0

Figure 10. Job Cards

The priority numbers are computed by adding the appropriate processing times. For instance, the priority number

of J-1-1 under Rule 1 is the sum, 3 + 5 + 2 = 10, of the processing times for J-1-1, J-1-2, and J-1-3. The priority number of J-1-1 under Rule 2 is the sum, 5 + 2 = 7, of the processing times for J-1-2 and J-1-3. Under each Priority Rule, the decision regarding the job to assign next is made by choosing that job having the largest priority number for the operation concerned.

The results of the computations using Priority Rules 1 and 2 are exhibited in Figure 11 where it is seen that Rule 2 results in the earlier completion of the three jobs.

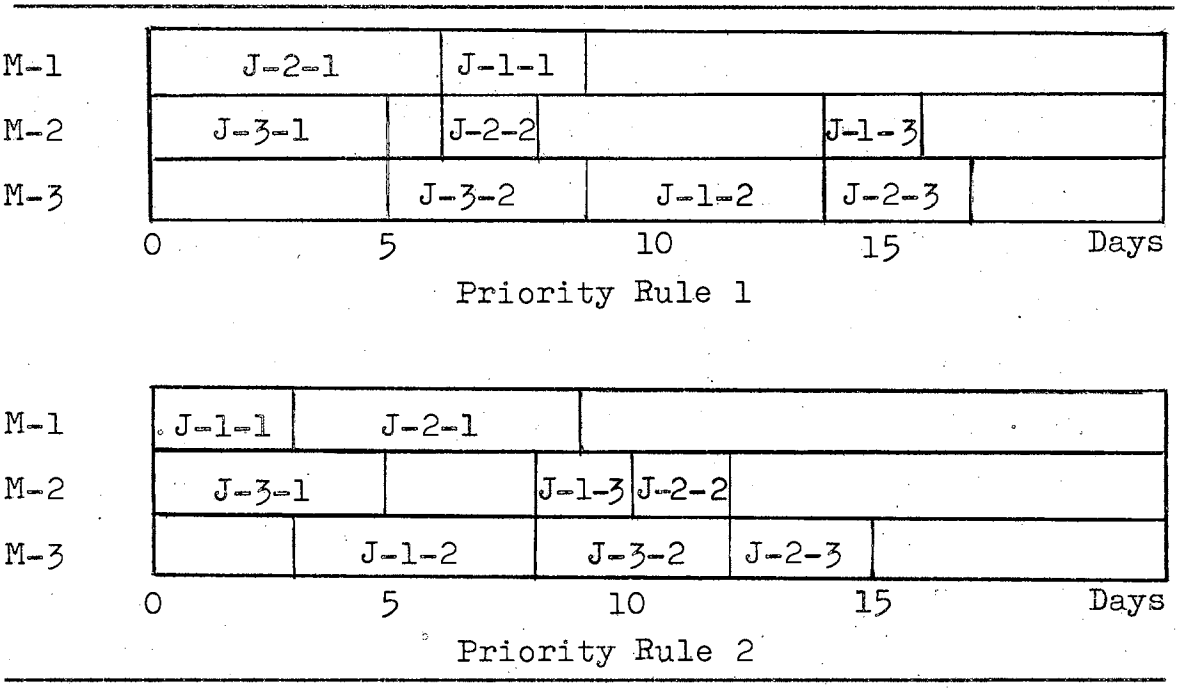

Figure 11. Performance of Priority Rules

Urgency Number Sequencing

Another interesting approach to the scheduling-sequencing problem has been developed by the Oklahoma State University Boeing Research Task Group [14]. This system provides a dynamic queue discipline that is a function of the total time remaining before due date for the order and the statistical properties of downstream flow time. As flow time conditions change in the system, the urgency number system will adapt itself by allocating scarce production time to those orders with the lowest probability of completion by their due dates. This scheme does not determine production capacity, but simply distributes the available capacity among the orders competing for the capacity in such a way that each has an approximately equal probability of being completed by its due date.

Regardless of the position of an order in the manufacturing system, there is a certain upstream history and a certain downstream future that has a bearing on the probability of completing the order by its due date. If the current date is designated C and the order due date D_i, then the time remaining before the due date for order i is $D_i - C$. For an order at machine center k, k = 1, k = 2, ..., k = n, the expected total flow time before completion will be

$$\sum_{j=k}^{n-k+1} \mu_j . \qquad (14)$$

The total flow time variance will be

$$\sum_{j=k}^{n-k+1} \sigma_j^2 . \tag{15}$$

The number

$$z_i = \frac{(D_i - C) - \sum_{j=k}^{n-k+1} \mu_j}{\sqrt{\sum_{j=k}^{n-k+1} \sigma_j^2}} \tag{16}$$

is a standardized variate on the distribution of total flow time, T_i. The order with the smallest algebraic value for z_i is the most urgent since z_i implicitly reflects the probability of completing order i by its due date.

The number z_i assigns an urgency number applicable to order i regardless of its upstream history or its position in the manufacturing system. It expresses the relative urgency in comparison with other orders in the queue based upon the time remaining and the statistical properties of the downstream flow time. However, it is noted that as $k \to n$, the distribution of T_i will deviate from normality. This is only important to the extent that the total flow time distributions for orders being compared differ. Of course, if t_j is distributed normally, T_i will be distributed normally regardless of the number of remaining downstream machine centers.

An order awaiting release to the manufacturing system

may be thought of as being in the queue of the first "machine" center, $k = 1$. This "machine" center would embrace a flow time made up from that series of acts required to initiate and complete release action for an order. After release, an order would be free to move to a real machine center. Release may be effected by choosing a specific release value, z^*, and releasing when $z_i < z^*$. Theoretically, z^* must be greater than zero if the condition of everincreasing queue length is to be avoided. The appropriate value for z^* must be empirically determined for a given situation by considering:

(a) The number of releases anticipated with $(D_i - C) < \sum_{j=1}^{n} \mu_j$ and the magnitude of each each inequality.

(b) The mix of orders in the system and the resulting idle time due to out of work conditions.

The urgency number model given by Condition (16) may be programmed for a digital computer. Required as inputs are current estimates of μ_j and σ_j^2 for all machine centers including the release process, due dates for each order, the routing for each order, and the current date. A computer run is made each day to determine an urgency number for all orders awaiting release to the manufacturing system and for all orders in process. The passage of one day, the availability of updated estimates of μ_j and σ_j^2, and the fact

that k will increase by one when the order flows through a machine center will alter the previously computed value of z_i.

Each order awaiting release or at machine center k has an assigned urgency number from the computer run. Each z_i implicitly reflects the probability of completing the order by its due date. Thus, z_i states which order should be worked first, which should be worked second, and so forth. Actually, the urgency numbers rank the downstream routes for orders in queue in accordance with their relative urgency. The algorithm gives precedence to those orders with the most critical route. This causes the allocation of scarce production time to fall to those orders which have the smallest implied probability of being completed by their due date.

Probability sequencing, as described above, is an expedite process that shifts scarce production time from those orders that will probably be completed before due date to those that have less probability of being completed before due date. The probabilities, although not explicitly specified, are implicitly reflected in the urgency numbers. The urgency number model is independent of any rigid assumption about the form of the distribution of total flow time, T_i, to the extent that the implied probabilities are comparable on a relative basis.

The Capacity Requirement Problem

The capacity requirement problem, as related to job shop operations, is defined as determining the level of capacity required at each machine center to optimize some criteria for performance. Selecting the performance criteria may vary in individual instances and was discussed earlier in this chapter.

Capacity can be measured in several ways; however, in this presentation, the units will be man-equipment hours per day. The three parameters that determine capacity level then are labor, equipment, and time. It should be emphasized that the relationship between these parameters is not always on a one-to-one basis; e.g., one man may be able to operate several machines simultaneously and similarly, some equipment may require the services of several men before they will function. In essence, capacity can be thought of as a time weighted value of labor and equipment working together in that relationship needed to perform the required function.

Since capacity is a function of these three variables, it is obvious that the capacity level in a job shop can be altered in a number of ways. Capacity adjustments can be made by the reallocation of the present work force to different machine centers. That is, some workers have multiple skills; e.g., a drill press operator may be able to operate a grinder. This type of capacity change adjusts

capacity levels at specific machine centers, but does not alter the over-all job shop capacity level. Capacity can also be changed by altering the labor force. That is, workers can be layed-off or hired. If additional workers are hired, equipment must be available for them to use. Other means of adjusting capacity levels include changing the number of hours worked per day and/or the number of days worked per week. In addition, work shifts can be adjusted up to a maximum of three 8-hour shifts per day. It is not the intention of this discussion to determine the economic trade-offs between these various methods of changing capacity, but only to show that these alternatives exist. The appropriate combination to use for a required capacity adjustment is an economic consideration based upon the cost of equipment, the various labor rates, and the forecasted demand for the particular machine center under consideration.

Intuition and Experience

A review of the literature indicates that very little work has been done in developing a method of handling the job shop capacity requirement problem. There has been some reference to this problem in context with larger problems of job shop control; however, there does not appear to be any appreciable amount of work designed to specifically handle this problem.

Since capacity level decisions are constantly being made in actual job shop situations, it is of interest to briefly examine how this problem is handled in most instances.

Interviews with job shop production managers indicate that past experience and intuition play a primary role in determining capacity levels at machine centers in most operating situations. The production manager is usually very familiar with the operating characteristics of the job shop and has a certain "feel" for the manner in which the orders will progress through the machine centers. He uses this "feel" in conjunction with rough estimates and past experience to predict future work loads and then matches capacity to these predictions. Capacity levels set in this manner are many times incorrect and cause a situation requiring constant readjustment on a short notice to correct for past mistakes. This situation often results in action being taken when, in fact, there is no need for corrective action, as well as the failure to initiate corrective action when it is required. Even though this atmosphere of panic is associated with job shop decisions, some production managers argue that there is no better way to operate the shop.

Linear Programming Approach

Dzielinski, Baker, and Manne [15] have developed a

linear programming model for solving the following production planning problem. Given a forecast of the demand for each product over a finite horizon of discrete planning periods, determine for all component parts the optimum number of parts to make on each manufacturing order and the planning period in which to place each order so that the total variable cost of operations is minimized. Although this is not the capacity requirement problem per se, it does answer some of the same questions. By examining this model, it is possible to gain some insight to a linear programming approach to the capacity requirement problem.

In the problem formulated by Dzielinski, et al. [15], the function of the linear programming computations is to select the optimal combination of production sequences for all of the parts in the system over the entire planning period.

The output of the linear programming computations is a set of production orders for the parts to be fabricated in order to satisfy the inventory and sales requirements. This computation indicates whether the required number of pieces for a part should be made in one batch, two batches, or more. It also determines the batch size(s) and the period in which the batch(es) should be made. These decisions are made so that the total discounted cost of future operations is minimized. The answers obtained from this model can be used to determine capacity requirements at machine centers;

however, only selected methods of adjusting capacity are considered. In addition, little attention is given to due date performance.

The unknowns, coefficients, and constants for the linear programming formulation are defined as follows:

Unknowns.

θ_{ij} - the fraction of the total requirement for the i^{th} part produced with the j^{th} alternative set-up sequence - $(i = 1, 2, \ldots, I)$, $(j = 1, 2, \ldots, J)$.

$W^1_{k\tau}$ - the number of workers assigned to first-shift operations on facility k during period $(t + \tau)$ without overtime - $(k = 1, 2, \ldots, K)$, $(\tau = 1, 2, \ldots, T)$.

$W^2_{k\tau}$ - the number of workers assigned to first-shift overtime operations on facility k during period $(t + \tau)$; each of these workers labors a fixed number of straight-time and overtime hours during the period.

$W^3_{k\tau}$ - the number of workers assigned to second-shift operations on facility k during period $(t + \tau)$ without overtime.

$W^4_{k\tau}$ - the number of workers assigned to second-shift overtime operations on facility k during period $(t + \tau)$; each of these workers labors a fixed number of straight-time and overtime hours during the period.

$D^+W_{k\tau}$ - the increase in the total number of workers employed at facility k from period $(t + \tau - 1)$ to period $(t + \tau)$.

$D^-W_{k\tau}$ - the decrease in the total number of workers employed at facility k from period $(t + \tau - 1)$ to period $(t + \tau)$.

Parameters recalculated each period.

$L_{ijk\tau}$ - the labor input required during period $(t + \tau)$ to carry out the j^{th} alternative set-up sequence on the k^{th} facility for part i.

$$L_{ijk\tau} = \begin{cases} 0 \\ a_{ik} + b_{ik}x_{ij\tau} \end{cases} \text{ when } x_{ij\tau} \begin{cases} = 0 \\ > 0 \end{cases}$$

where a_{ik} and b_{ik} refer, respectively, to the standard labor set-up time and the standard unit running time for the i^{th} part on the k^{th} facility. The numbers $x_{ij\tau}$ refer to the amount of

part i required by sequence j during period τ. These numbers define the alternative production sequences.

Constants.

W_k - the maximum number of workers that can be assigned to facility k during a single shift.

H^1 - the total number of first-shift hours per period, excluding overtime.

H^2 - the total number of first-shift hours per period, including a fixed amount of overtime.

H^3 - the total number of second-shift hours per period, excluding overtime.

H^4 - the total number of second-shift hours per period, including a fixed amount of overtime.

$R^1_{k\tau}$ - the first-shift wage for facility k, without overtime, discounted over τ periods.

$R^2_{k\tau}$ - the first-shift wage for facility k, with overtime, discounted over τ periods.

$R^3_{k\tau}$ - the second-shift wage for facility k, without overtime, discounted over τ periods.

$R^4_{k\tau}$ - the second-shift wage for facility k, with overtime, discounted over τ periods.

$\Gamma_{o\tau}$ - the cost of laying off one worker, discounted over τ periods.

$\Gamma_{h\tau}$ - the cost of hiring one worker, discounted over τ periods.

$c_{i\tau}$ - the unit material cost of part i, discounted over τ periods.

The linear programming problem is stated as follows:

$$\min \sum_{\tau=1}^{T} \sum_{k=1}^{K} \sum_{i=1}^{I} \left\{ \sum_{r=1}^{4} \left[R^r_{k\tau} W^r_{k\tau} + \Gamma_{o\tau} D^- W_{k\tau} + \Gamma_{h\tau} D^+ W_{k\tau} \right] + \sum_{j=1}^{J} \theta_{ij} c_{i\tau} x_{ij\tau} \right\} \tag{17}$$

subject to:

$$\sum_{j=1}^{J} \theta_{ij} = 1, \qquad i = 1, \ldots, I \tag{18}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{J} L_{ijk\tau} \theta_{ij} \leq \sum_{r=1}^{4} H^r W^r_{k\tau}, \qquad \begin{cases} k = 1, \ldots, K \\ \tau = 1, \ldots, T \end{cases} \tag{19}$$

$$\sum_{r=1}^{4} W^r_{k\tau} = \sum_{r=1}^{4} W^r_{k,\tau-1} + D^+ W_{k\tau} - D^- W_{k\tau}, \qquad \begin{cases} k = 1, \ldots, K \\ \tau = 1, \ldots, T \end{cases} \tag{20}$$

$$W^1_{k\tau} + W^2_{k\tau} \leq W_k, \qquad \begin{cases} k = 1, \ldots, K \\ \tau = 1, \ldots, T \end{cases} \tag{21}$$

$$W^3_{k\tau} + W^4_{k\tau} \leq W_k, \qquad \begin{cases} k = 1, \ldots, K \\ \tau = 1, \ldots, T \end{cases} \tag{22}$$

$$\theta_{ij} \ , \ W^{r}_{k\tau} \ , \ D^{+}W_{k\tau} \ , \ D^{-}W_{k\tau} \ , \ \text{all} \geq 0. \qquad (23)$$

There are I equations in Condition (18). These restrictions specify that the total planned requirements for each part must be satisfied by a convex combination of the admissible production sequences.

There are KT inequations in Condition (19). These inequations insure that the total capacity of machine group k during period τ will be sufficient to produce the assigned work load.

There are KT equations in Condition (20). These are simple balance equations that relate the size of the work force from one period to the next. The initial work force availability is predetermined prior to each lot-size programming calculation.

There are KT inequations in both Conditions (21) and (22). These inequations limit the number of workers who can be assigned to machine group k in period τ for both the first and second shifts.

Queueing Theory Approach

It has been suggested that queueing theory might offer an approach to the job shop capacity requirement problem. This approach has proved quite useful in determining capacity requirements when a single operation is to be performed; e.g., collecting tolls on a highway or checking out customers in a grocery store. When this type situation exists,

then queueing theory can be applied quite successfully to determine the number of toll booths that should be in operation or the number of check-out stations that should be available to optimize some performance criteria. However, in a job shop operation, the situation is not quite as simple. Instead of a single operation being performed on each incoming item, there is an entire sequence of operations that must be performed. Moreover, this sequence is usually different for each incoming item. Because of this compounding of stations affect and the lack of uniformity in the sequence of stations, the mathematical formulation in terms of queueing theory has not evolved at present. However, further work in this area may result in a practical solution to the job shop capacity requirement problem in terms of queueing theory.

CHAPTER III

THE CAPACITY REQUIREMENT ALGORITHM

The purpose of any algorithm is to set forth a computational procedure for accomplishing some objective. The capacity requirement algorithm's objective is to establish a statistically based solution to the capacity requirement problem in a job shop operation. This algorithm is not designed to provide an exact optimum solution to the problem. It is an attempt to provide a practical solution with improved performance characteristics.

The following criteria are employed to evaluate the performance of the algorithm: productive time, overtime (type I and II), idle time, backlog, in-process inventory, efficiency, and completion date performance. Productive time is the time the machine centers are in operation and is equal to basic capacity plus overtime minus idle time. There are two types of overtime built into the algorithm, type I and type II. Type I overtime might refer to hours in excess of an eight-hour work-day while type II overtime could specify work performed on weekends or holidays. Idle time is that time when no work is available at a machine center. Backlog is the machine time of orders in the queue at a machine center. In-process inventory is the machine

time for orders that have been released from the pool of uncommitted work, but are not yet completed. Efficiency is the ratio of productive time to the sum of basic capacity and overtime. The mean completion date performance, μ, is the mean number of days past due date for all orders passing through the system. The standard deviation of completion date performance, σ, is expressed in days.

Theory of the Algorithm

The capacity requirement algorithm is an inductive statistical method in that it uses a small body of data to make generalizations about a larger system of similar data. These generalizations are in the form of estimates or predictions. Before describing the algorithm, however, it is important to see how it conforms to the theory of inductive statistical methods.

The concepts of a population and a sample are basic to inductive statistical methods. Any finite or infinite collection of individual objects or events constitutes a population. A population is not thought of as just a group of things specified by numbering them, but rather as an aggregate determined by some property that distinguishes between things that do and things that do not belong. In contrast, a sample, defined as a portion of a population, has the connotation of incompleteness.

In the capacity requirement algorithm, the population

under consideration is the repeated estimates of orders that will arrive at a given machine center on some future date. These estimates are based on the mean flow times of the machine centers through which the orders must pass before reaching the particular machine center under consideration. The concept of flow times was discussed in Chapter II. The sample from this population consists of the estimated daily arrival of orders at the specified machine center over a planning period of θ days. The magnitude of θ must be determined outside the algorithm and is a function of the mean flow time estimates; i.e., better mean flow time estimates make it possible to have longer planning periods.

Another important concept of inductive statistical methods is that of a distribution. The fact that some characteristics of individuals of a population are not the same for every individual leads immediately to the recognition of a distribution for these characteristics. This distribution of some particular property of the individuals in a population is a collective property of the population. In addition, the average and other characteristics of the distribution are also collective properties of the population. The methods of inductive statistics provide the means for learning about such population characteristics from a study of samples. These methods are based upon the mathematical properties of sampling distributions of sample statistics such as the sample average and the sample range.

The characteristic of interest in the capacity requirement algorithm is the machine time (operation plus setup) of each order that arrives at the specified machine center. Since the orders that pass through any machine center are of a wide variety, their machine times do, in fact, form a distribution of values. The average daily arrivals over the planning period θ is the sample statistic used in the algorithm.

If it were practical or possible to examine an entire population, that population could be described by using whatever numbers, figures, or charts resulted from the investigation. However, since it is ordinarily inconvenient or, in the case of the capacity requirement algorithm, impossible to observe every item in the population, a sample is taken. The task is then to generalize from a sample to the whole population. Such generalizations about characteristics of a population from a study of one or more samples from the population are termed statistical inferences.

Statistical inferences take two forms: estimates of the magnitudes of population characteristics and tests of hypotheses regarding population characteristics. Both are useful for determining which among two or more courses of action to follow in practice when the correct course is determined by some particular, but unknown, characteristic of the population.

Statistical inferences all involve reaching conclusions

about population characteristics from a study of samples which are known or assumed to be portions of the population concerned. Statistical inferences are basically predictions of what would be found to be the case if the parent population were fully analyzed with respect to the relevant characteristics.

Both forms of statistical inferences are used in the capacity requirement algorithm. Estimates of the size of average daily work loads arriving at a machine center are made in order to establish the machine center's basic capacity level. Once this basic capacity level is established, tests of hypothesis are made to determine when a change occurs in the magnitude of average daily arrivals.

In order to be able to make inferences of a substantial character, the nature of the sampling operation must be known. That is, a hypothetical population of drawings needs to be defined. The statistical inferences made will be rigorous if, and only if, the inductive technique used is appropriate to the sampling procedure actually employed. In other words, in a strict sense, statistical inferences can only be made with respect to the hypothetical population of drawings defined by the sampling operation concerned. It is important to use a sampling procedure in which the relevant parameters of the population of drawings bear a known relation to the corresponding parameters of the real life situation.

If a sampling scheme is to provide a valid basis for inferences, it is necessary that the selection of the individuals to be included in a sample involve some type of random selection; that is, each possible sample must have a fixed and determined probability of selection. The most widely used type of random selection is simple (or unrestricted) random sampling. For a sampling scheme to qualify as a simple random sample, it is not sufficient that each individual in the population have an equal chance of appearing in the sample, as is sometimes said, but it is sufficient that each possible sample have an equal chance of being selected. It must be emphasized that the randomness of a sample is inherent in the sampling scheme employed to obtain the sample and is not an intrinsic property of the sample itself.

A particular sample often qualifies as a sample from any one of several populations. For example, a sample of n items from a single carton is a sample from that carton, from the production lot of which that carton is a portion, and from the production process concerned. By drawing these items from the carton in accordance with a simple random sampling scheme, it can be insured that they are a simple random sample from the carton, not from the production lot or the production process. Only if the production process is in a state of statistical control, may the sample also be considered a simple random sample from the

production lot and the production process. In a similar fashion, a sample of repeated estimates of future work loads can be validly considered a random sample from the conceptually infinite population of estimated future work loads by the same procedure, only if the estimating procedure is in a state of statistical control. A random sample cannot be drawn from this population by mechanical randomization, so it must be recognized that there is a random sample only by assumption. This assumption is warranted if previous data indicate that the estimating procedure is in a state of statistical control; unwarranted if the contrary is indicated; and a leap in the dark if no previous data are available.

It is important, in practice, to know from which of several possible parent populations a sample was obtained. This population is called the sampled population, and may be quite different from the population of interest, called the target population, to which it is desired to have the conclusions of the analysis be applicable. In practice, they are rarely identical, though the difference is often small. The further the sampled population is removed from the target population, the more the burden of validity of conclusions is shifted from the shoulders of the statistician to those of the subject matter expert, who must place greater reliance on "other considerations."

The basic assumption underlying most statistical

techniques is that the data are a random sample from a stable probability distribution, which is another way of saying that the process is in statistical control. It is the validity of this basic assumption that the control chart is designed to test. Although these assumptions cannot be rigorously defended on a theoretical basis in the case of the capacity requirement algorithm, empirical evidence, given in Chapter IV, shows that the performance characteristics are improved with proper selection of parameter values. This fact alone lends validity to the basic assumptions.

Concept of the Algorithm

The concept of the capacity requirement algorithm can be best explained by comparing it with a well-know inductive statistical method used in quality control. In a typical quality control situation, one of the primary objectives is to provide a method for maintaining the production process at some predetermined level of performance; i.e., to establish a basis for current decisions during production regarding when to take corrective action and when to continue operations.

When some form of process average is chosen as the control variable, an $\bar{x}$ chart becomes an appropriate control device. This chart is constructed in the following manner. Several observations are taken of the parameter in question. These observations are averaged to obtain an $\bar{x}$ value. After

a number of these $\bar{x}$ values have been calculated, they are averaged to form an $\bar{\bar{x}}$ value. This $\bar{\bar{x}}$ value is established as the process mean; i.e., the desired level of performance for the process.

Upper and lower control limits, UL and LL, are established about $\bar{\bar{x}}$ to indicate the acceptable range of process performance. A control chart showing $\bar{\bar{x}}$ and the control limits then can be constructed. Values for $\bar{x}$ are continually plotted on this chart and, if $LL < \bar{x} < UL$, the process is considered to be in statistical control. That is, the process mean has not changed significantly from $\bar{\bar{x}}$. When an $\bar{x}$ value falls outside the control limits, there is reason to believe that the process mean has shifted. When this occurs, measures are taken to locate the cause of variation and to bring the production process back to the predetermined level of performance.

The capacity requirement algorithm provides an analogous method for monitoring capacity levels at machine centers in a job shop operation. The capacity requirement problem is concerned with providing adequate capacity levels at machine centers to process the orders that require the use of these machine centers. The parameter chosen for control purposes is $\bar{x}_{k\theta_\varphi}$, the estimated average daily work load that is located at the machine center k over the following planning period of θ days. After @ values for $\bar{x}_{k\theta_\varphi}$ have been calculated, they are averaged to form

an $\bar{\bar{x}}_{k\theta_\varphi}$ value. This $\bar{\bar{x}}_{k\theta_\varphi}$ value is adjusted to reflect machine backlog conditions and established as the desired capacity level at the machine center.

Upper and lower control limits, UL_k and LL_k, are established about $\bar{\bar{x}}_{k\theta_\varphi}$ to indicate the range of expected average daily work loads that may occur at the machine center by chance variation rather than as a result of an actual change in the average daily work loads. Values for $\bar{x}_{k\theta_\varphi}$ are calculated daily and checked against the control limits. When a value for $\bar{x}_{k\theta_\varphi}$ falls outside the control limits, there is reason to believe that a shift has occurred in the average daily work load arriving at the machine center. When this situation occurs, it is desirable to establish a new machine capacity level to reflect the shift in average daily work loads. A new value for $\bar{x}_{k\theta_\varphi}$ is calculated by averaging the preceding Θ values for $\bar{x}_{k\theta_\varphi}$ and adjusting for backlog conditions. The machine center capacity level is then adjusted to this new value for $\bar{\bar{x}}_{k\theta_\varphi}$. New values are also calculated for the control limits, UL_k and LL_k, and the system is ready for operation as described previously.

As evidenced by the preceding discussion, there are two primary distinctions between these two control procedures. First of all, in quality control, actual current observations are used, while the capacity requirement algorithm uses estimates of future occurrences. The second

major distinction is more basic in nature. In quality control, when an observation falls outside the control limits, adjustments are made to bring the production process back to the predetermined level of performance. In capacity control, when an observation falls outside the control limits, no attempt is made to alter the anticipated average daily work load, but rather to adjust the capacity level at the machine center to accommodate the new anticipated average daily work load.

This decision process for setting and adjusting capacity levels at machine centers is established in an effort to improve the operating characteristics of machine centers in a job shop operation.

Definition of Constants and Variables

The following constants and variables are required by the capacity algorithm:

μ_j - An estimate of the average flow time required for an order to be processed through machine center j. This estimate is updated periodically to reflect the dynamic nature of the job shop situation. It may be calculated in several ways. First, it may be based entirely on the average of past flow times of orders that have passed through the j^{th} machine center. Second, it may be based entirely on the estimated

average flow time of orders that are currently in queue at the machine center. Finally, it may be based on some combination of the two; e.g., 30 per cent past history and 70 per cent current estimates.

μ_{ik} - An estimate of the average flow time of order i on its arrival at machine center k.

$$\mu_{ik} = \sum_{j=1}^{k-1} \mu_j \qquad (24)$$

where k - 1 refers to those particular machine centers that order i must pass through prior to arriving at machine center k.

h_{ik} - The operation plus setup time required by order i at machine center k. This value is known or can be estimated from work sheets, standard data, etc.

θ - The length of the planning period in time units. For convenience, a time unit will be considered one day. The planning period refers to the future period of time over which estimates of average daily work loads will be made. The value of θ used in the system is specified, but may be changed for different simulation runs.

Θ - The number of observations of $\bar{x}_{k\theta_\varphi}$ used in computing $\bar{\bar{x}}_{k\theta_\varphi}$. The value of Θ used in the system is specified, but can be changed for different simulation runs.

t - The positive number exceeded by $100(\frac{\alpha}{2})$ per cent of the t distribution with $\Theta - 1$ degrees of freedom, where α is the desired confidence level. The value for t used in the system is specified, but may be changed for different simulation runs.

$x_{k\theta_\varphi}$ - An estimate of the total work load, in time units, that will arrive at machine center k during the following planning period of θ days as calculated on day φ. This value is determined by the algorithm in the following manner. Each order i is examined to see if its value for μ_{ik} falls within the following planning period of θ days. If it does, the value h_{ik} is included in $x_{k\theta_\varphi}$.

$$x_{k\theta_\varphi} = \sum_{i=0}^{m} h_{ik} \qquad (25)$$

where m refers to those orders whose μ_{ik} lies within θ. Estimates are made for all orders, including those presently on the floor, those in the pool of unreleased work,

and those not yet a part of the system.

$\bar{x}_{k\theta_\varphi}$ - An estimate of the average daily work load that is located at machine center k over the following planning period of θ days as calculated on day φ.

$$\bar{x}_{k\theta_\varphi} = \frac{x_{k\theta_\varphi}}{\theta} + f(h_{ik}) \qquad (26)$$

where $f(h_{ik})$ is a function of the current backlog of work at machine center k.

$\bar{\bar{x}}_{k\theta_\varphi}$ - The average of the estimated average daily work loads that arrive daily at machine center k over the past Θ days as calculated on day φ, plus $g(h_{ik})$, an adjustment for work backlog at the machine center.

$$\bar{\bar{x}}_{k\theta_\varphi} = \Big(\sum_{\gamma=-\Theta}^{-1} \bar{x}_{k\theta_{\varphi+\gamma}}\Big)\Big/\Theta + g(h_{ik}). \qquad (27)$$

$S_{\bar{x}_{k\theta_\varphi}}$ - An estimate of the standard deviation of the estimated average daily work loads at machine center k as calculated on day φ.

$$S_{\bar{x}_{k\theta_\varphi}} = \sqrt{\frac{\sum_{\gamma=-\Theta}^{-1}\Big[\bar{x}_{k\theta_{\varphi+\gamma}}+f(h_{ik})\Big]^2-\Big[\sum_{\gamma=-\Theta}^{-1}\bar{x}_{k\theta_{\varphi+\gamma}}+f(h_{ik})\Big]^2\Big/\Theta}{\Theta-1}}. \qquad (28)$$

UL_k - The upper limit of the confidence interval at machine center k.

$$UL_k = \bar{\bar{x}}_{k\theta_\varphi} + t\, S_{\bar{x}_{k\theta_\varphi}} / \sqrt{\Theta} . \qquad (29)$$

LL_k - The lower limit of the confidence interval at machine center k.

$$LL_k = \bar{\bar{x}}_{k\theta_\varphi} - t\, S_{\bar{x}_{k\theta_\varphi}} / \sqrt{\Theta} . \qquad (30)$$

Many of these variables can be calculated in several ways; e.g., weighted averages can be used instead of simple averages. Each of the formulas given in this section indicates only one of the possible ways that the variable in question may be obtained.

The Algorithm

Using the constants and variables as defined in the previous section, the capacity requirement algorithm is given as follows:

a. Calculate daily $\bar{x}_{k\theta_\varphi}$ for each machine center for Θ days.

b. At the end of Θ days, calculate $\bar{\bar{x}}_{k\theta_\varphi}$ for each machine center.

c. Set the capacity level at each machine center at $\bar{\bar{x}}_{k\theta_\varphi}$.

d. Construct the confidence limits UL_k and LL_k for each machine center.

e. Continue to calculate $\bar{x}_{k\theta_\varphi}$ daily for each machine center and check against the limits UL_k and LL_k.

f. If $UL_k < \bar{x}_{k\theta_\varphi} < LL_k$, calculate a new value for $\bar{\bar{x}}_{k\theta_\varphi}$ on the basis of the past Θ observations of $\bar{x}_{k\theta_\varphi}$.

g. Adjust the capacity level at the machine center to the new value for $\bar{\bar{x}}_{k\theta_\varphi}$.

h. Calculate new values for the confidence limits UL_k and LL_k.

i. Return to step e.

CHAPTER IV

SIMULATION ANALYSIS

An analysis is imperative if worthwhile results are to evolve from an experiment. The analysis of the experiments with the capacity requirement algorithm will consist of an explanation of the computer program model used to produce the test cases, a discussion of the performance criteria used to judge the algorithm's output, and the resulting effect and relative importance of the algorithm's parameters.

Description of the Model

The computer program model used to test the capacity requirement algorithm was developed by the Oklahoma State University Operations Research Group. A listing of this program is included as Appendix A.

The model simulates a ten machine center job shop in which orders are released from a pool of uncommitted work under the Urgency Number Sequencing System described in Chapter II. Orders are released to the shop floor when the probability of completion by their due date is .50. The orders may require from one to ten machine operations and may pass through the same machine center more than

once. A random order generator subroutine governs the order review time, the rate of orders being produced, the number of operations per order, the maximum number of orders generated, the machine time per operation, and the initial system load. The same orders are generated on the same days for each of the nine test cases under consideration. The simulation time represents 110 days of operation.

The model contains a number of constants which allows it to be adjusted in many ways, particularly the manner in which machine center flow time means and variances are calculated. Some explanation of these constants will add understanding to the effort that was devoted to making the model as versatile as possible.

The constant WBAR modifies anticipated arrivals at machine centers by taking into consideration the amount of machine time in the queue for orders with a negative z number. This expression represents the backlog consideration, $f(h_{ik})$, in Condition (26). The value used for WBAR in the test cases is .10. In other words, the estimated average daily work load at a machine center includes 10% of the machine time of the current backlog of orders with negative z numbers at that machine center.

The constants W1, W2, and W3 determine the population or orders whose machine center flow time values are used to calculate the flow time means and variances for the machine centers. This concept is discussed in the section

on flow time relationships in Chapter II.

The constant W1 indicates the value of machine center flow times of orders that have previously entered a machine center, when calculating that machine center's flow time mean and variance. The value for W1 in the test cases is .70. This indicates that a weight of 70% is given to the machine center flow times for orders that have previously been processed at a machine center, in arriving at that machine center's flow time mean and variance; i.e., 70% of the population of orders whose machine center flow times are used to calculate each machine center's flow time mean and variance, comes from orders that were in that machine center at some prior time.

The constant W2 indicates the value of the machine center flow times of the orders in the immediate backlog of work at each machine center, which is used to calculate that machine center's flow time mean and variance. The value for W2 in the test cases is .30. This indicates that the machine center flow times of orders in the current backlog of the machine center, carries a weight of 30% in establishing the value of that machine center's flow time mean and variance; i.e., 30% of the population of orders whose machine center flow times are used to calculate each machine center's flow time mean and variances comes from orders that are currently in the queue of that machine center.

A third constant, W3, reflects the value that the

average machine center flow time of all orders in the system represents when calculating a particular machine center's flow time mean and variance. For the test cases, W3 is set at zero. This value was chosen because this adjustment was not considered significant initially. However, the ability to let the total work in the system have some significance in calculating machine center flow time means and variances is provided to make the model as versatile as possible.

The constant TAPER determines the extent to which each proceeding day's calculations of the estimated average daily work load at a machine center, $\bar{x}_{k\theta_\varphi}$, contributes to the value calculated for the basic capacity level at the machine center, $\bar{\bar{x}}_{k\theta_\varphi}$. That is, the most current data is weighted more heavily with lesser significance given to data as it gets older.

In addition to the constants of the model, the capacity requirement algorithm contains several constants that define its operation.

The constant KAPIN establishes the minimum increment by which capacity at a machine center can be increased or decreased. The value selected for test cases one through eight is one-half man-machine day; however, case nine, which exhibits constant capacity, has KAPIN set equal to zero. Setting KAPIN at zero keeps the basic capacity level at the machine centers from ever changing.

The constant KAPMX determines the maximum amount that

capacity at a machine center can be increased on a given day. The value for the test cases is three man-machine days.

The constant KAPDE establishes the maximum amount that capacity at a machine center can be decreased on a given day with the restriction, of course, that the machine center actually has that amount of capacity. No machine center can have a negative capacity. The value selected for the test cases is three man-machine days.

The constant INTZ establishes the smallest increment of the order z number used to sequence orders in the machine center queue. The value selected for the test cases is one-tenth. In other words, the sequence of orders in the queue at a machine center would not change until the z number of an order is at least one-tenth smaller than the z number of the order in front of it. At this point, the orders would be resequenced, the order with the smallest z number being first.

The constant ZOVT1 establishes the machine center z number at which type 1 overtime is employed. The value selected for the test cases is -0.50.

The constant ZOVT2 establishes the machine center z number at which type 2 overtime is employed. The value selected for the test cases is -1.00.

The constant MOTR1 establihses the maximum amount of type I overtime that can be employed at a machine center. The value selected for the test cases is 50% of the basic

capacity of the machine center in question.

The constant MOTR2 establishes the maximum amount of type II overtime that can be employed at a machine center. The value selected for the test cases is also 50% of the basic capacity of the machine center in question.

In addition to the constants of the capacity requirement algorithm, different values of three parameters are selected for special analysis in the test cases. The term NXBAR in the program model is θ in the theoretical presentation of the algorithm; i.e., the length of the planning period. The term NXDBL is Θ in the theoretical presentation of the algorithm; i.e., the number of observations of $\bar{x}_{k\theta_\varphi}$ used to establish $\bar{\bar{x}}_{k\theta_\varphi}$. The term TLOWR is the value of t in standard deviations used to calculate the lower control limit in the theoretical presentation. The term TUPPR is the value of t in standard deviations used to calculate the upper control limit in the theoretical presentation. Actually, when talking about TLOWR and TUPPR, the parameter under consideration is the confidence level, α. However, α is dependent upon both t and Θ. Therefore, since Θ is already specified, values of t are selected to correspond with the levels of α that are to be tested.

Performance Criteria

As discussed in Chapter II, there are a number of criteria that can be used to measure the effectiveness of any decision rule under consideration. For this study,

the following were selected as performance criteria for the capacity requirement algorithm: productive time, overtime I and II, idle time, backlog, in-process inventory, efficiency, and completion date performance. Total order flow time is considered an important criteria; however, the program model did not calculate these values.

The units of productive time, overtime I and II, idle time, backlog, and in-process inventory are tenths of days for the sum of all ten machine centers for a period of 110 days of operation of the job shop. Efficiency is the ratio of the time used to the time available for all ten machine centers and is expressed as a per cent. The mean completion date performance, μ, is expressed as the average number of days past the order due date for all completed orders, and the standard deviation of completion date performance, σ, is expressed in days.

Table V shows the individual values for each effectiveness criteria for each test case.

Test Case Evaluation

The following evaluation is directed towards two objectives. First of all, the adjustment of basic capacity under the capacity requirement algorithm is compared to the constant basic capacity situation. Secondly, the effect and relative importance of θ, Θ, and α on the various performance criteria are examined.

Capacity Requirement Algorithm Versus Constant Basic Capacity

A cursory review of Table V shows that the constant basic capacity case possesses a better set of performance effectiveness results than some of the cases employing the capacity requirement algorithm and a worse set than other cases. A moment's reflection indicates that this is a reasonable result. The capacity requirement algorithm provides the job shop manager with the opportunity to improve his operations, provided he judiciously selects values for the algorithm's parameters. A poor choice of values for the parameters will result in a performance record below that which would have occurred if the manager had never changed his basic capacity. A simple exercise will also point out this fact. Circle the best value among the nine test cases for each effectiveness criteria. Likewise, underline the worst value among the test cases for each effectiveness criteria. Now, notice that none of the effectiveness criteria values for the constant basic capacity case are either circled or underlined, which again shows that the values of the parameters are most important if the capacity requirement algorithm is going to be used in lieu of a decision rule which does not allow basic capacity to change.

Importance of Parameters

At this point, each of the parameters (θ, Θ, and α)

TABLE V

SIMULATION RESULTS

Case	Parameters	Productive Time (10^{ths} of days)	Overtime Type I (10^{ths} of days)	Overtime Type II (10^{ths} of days)	Idle Time (10^{ths} of days)	Backlog (10^{ths} of days)	In-Process Inventory (10^{ths} of days)	Efficiency (%)	Completion Date Performance μ (days)	Completion Date Performance σ (days)
1	$\theta = 5$ $\Theta = 30$ $\alpha = .10$	12,055	1,805	233	246	3,716	13,163	98.00	1.8	1.5
2	$\theta = 15$ $\Theta = 30$ $\alpha = .10$	12,430	156	1	3,343	2,757	9,031	78.81	6.8	4.0
3	$\theta = 5$ $\Theta = 15$ $\alpha = .10$	12,043	1,810	246	236	3,617	13,233	98.08	1.5	1.3
4	$\theta = 15$ $\Theta = 15$ $\alpha = .10$	12,518	26	-	3,553	2,616	8,462	77.89	7.4	3.8
5	$\theta = 5$ $\Theta = 30$ $\alpha = .20$	12,047	1,834	263	250	3,678	13,164	97.97	1.8	1.5
6	$\theta = 15$ $\Theta = 30$ $\alpha = .20$	12,414	165	1	3,416	2,726	9,024	78.42	6.7	4.1
7	$\theta = 5$ $\Theta = 15$ $\alpha = .20$	12,040	1,826	244	189	3,638	13,189	98.45	1.5	1.2
8	$\theta = 15$ $\Theta = 15$ $\alpha = .20$	12,508	30	-	3,538	2,604	8,460	77.95	7.4	3.8
9	Constant Basic Capacity	12,103	1,486	462	845	3,653	12,813	93.47	2.4	2.3

will be analyzed and their relative importance and effect on performance criteria determined.

Importance of θ. Looking once again at Table V, a number of patterns begin to form. The most obvious of these patterns is that the corresponding values for each performance criteria for cases 1, 3, 5, and 7 are of similar magnitude. Likewise, the corresponding values for each performance criteria for cases 2, 4, 6, and 8 are of similar magnitude, the former set of cases being easily distinguished from the latter set irrespective of the performance criteria used. An examination of the changes in the values of the parameters in these cases show that the value of θ was constant at 5 for cases 1, 3, 5, and 7 and constant at 15 for cases 2, 4, 6, and 8. This fact would indicate that the length of the planning period, θ, is the most important of the three parameters under consideration. As θ is increased, at least within the range of values tested, improvement is observed in the following performance criteria: productive time, overtime, backlog, and in-process inventory. However, an adverse effect is observed in idle time, efficiency, and completion date performance.

This behavior can be explained as follows. As the planning period is increased in scope, basic capacity levels reflect more the long term demands upon the system. Productive time is larger because basic capacity is more stable at a level which better describes the system's

demand for production capability. Less overtime is required because the requirement for additional capacity is reflected in the basic capacity capability and not in the overtime capacity capability. Because more of the production capability is vested in basic capacity, backlog levels are not as great. Likewise, it follows that in-process inventories are also reduced.

On the other hand, a system that is less responsive to short term changes in demand because of a long range planning period, experiences some difficulties. Since more production capability is vested in basic capacity and cannot be quickly reduced, it is reasonable to expect that the amount of idle time in the system will increase. Additionally, recalling that the efficiency figure is the ratio of time used to time available, it is not unreasonable to expect efficiency to decrease as basic capacity levels are increased and the same amount of work is released to the system.

A paradoxical situation is experienced in connection with the completion date performance. Although basic capacity is higher, the backlog is less, and in-process inventory is lower, the completion date performance is adversely affected by an increase in the planning period. Reflection upon this situation, however, does give some possible explanations. Remembering that type I overtime is not employed at a machine center until that machine center's z number is less than -0.50, it is quite possible

that orders could be progressing through the machine centers fast enough that overtime is not employed, but not quickly enough to exhibit exceptionally good completion date performance. On the other hand, when the basic capacity levels at the machine centers are relatively low, overtime capacity is employed when critically needed which considerably speeds up orders in danger of missing their due dates. The overtime capacity, in effect, acts like a shot in the arm to a system whose basic capacity levels reflect only short term demands upon the system. This added boost aids in completion date performance but does not provide relief for an overcrowded system. Adjustments in the triggering mechanism for more timely addition of overtime might improve completion date performance in the cases where longer planning periods are employed.

Importance of Θ. The importance of Θ, the number of observations of $\bar{x}_{k\theta_\varphi}$ used in computing $\bar{\bar{x}}_{k\theta_\varphi}$, is not quite as obvious as the importance of the planning period, θ. A comparison of the corresponding performance criteria of cases 1 and 3 where the values of θ and α are constant at 5 and .10, respectively, and Θ changes from 30 to 15, shows no significant difference. However, once the level of either θ or α is raised, changes in Θ are reflected in some of the performance criteria.

For example, consider a comparison of cases 2 and 4. The values of θ and α are constant at 15 and .10, respectively, and Θ is changed from 30 to 15. There is no

significant change in productive time or efficiency; however, the other measures do exhibit a significant change (a minimum of 5%). The really significant change occurs in the reduction of overtime. In this case comparison, when Θ is decreased 50%, overtime decreases 83%. This observation is reasonable because, with a small Θ, basic capacity can be quite responsive to the current time frame, precluding the occasion for overtime to be required.

The reasons for the increase in idle time, decrease in backlog, decrease in in-process inventory, and decrease in completion date performance are postulated to be the same as those presented in the preceding section concerning θ.

Another observation of the effect of changes in Θ, ceteris paribus, is seen by comparing cases 5 and 7 where θ and α are 5 and .20, respectively, and Θ is changed from 30 to 15. In this comparison, the value of α, though constant, is at a higher level than in the two previous case comparisons. There is no significant difference in productive time, type I overtime, backlog, in-process inventory, or efficiency. There is, however, significant improvement in type II overtime, idle time, and completion date performance.

The decrease in overtime observed in the comparison of cases 5 and 7 is not observed in the comparison of cases 1 and 3 which would indicate that the change is due

to α rather than Θ.

The significant decrease in idle time is noticed to a lesser degree in the comparison of cases 1 and 3. The higher level of α, however, tends to emphasize this change. This observation indicates that both decreases in Θ and increases in α cause idle time to decrease. However, α appears to be the more important factor.

The improvement in completion date performance can best be explained in terms of the significantly reduced idle time. The fact that machine center utilization (efficiency) improves slightly with relatively high rates of overtime, however, also accounts for some improvement in completion date performance.

A final observation of the effect of changes in Θ, ceteris paribus, is observed by comparing cases 6 and 8 when θ and α are 15 and .20, respectively, and Θ changes from 30 to 15. There is no significant difference in productive time, idle time, backlog, or efficiency. There is, however, a significant decrease in overtime, in-process inventory, and completion date performance.

A similar improvement in overtime is noticed in the comparison of cases 2 and 4. This fact suggests that decreases in Θ play a secondary role in decreasing overtime; the primary factor in decreasing overtime is an increase in θ.

The decrease in in-process inventory is also noticed in the comparison of cases 2 and 4. This fact substantiates

that the value of θ plays the primary role in determining the level of in-process inventory. As θ increases, in-process inventory decreases. Changes in Θ on the other hand, play a secondary role. As Θ decreases, in-process inventory decreases.

The decrease in completion date performance is attributed to the paradoxical situation concerning the use of overtime described in the discussion of the importance of θ. In summary, the effects of changes in Θ are categorized in the following manner. There is no significant effect on productive time regardless of the values of θ and α. When both θ and α are small, decreases in Θ increase overtime. When θ is large, the magnitude of overtime is reduced considerable, and decreases in Θ cause overtime to decrease over all values of α tested. When θ is small and α large, decreases in Θ also cause decreases in overtime but to a lesser degree. When θ and α are both small or both large, changes in Θ do not have a significant effect on idle time. This fact would indicate that θ and α affect idle time in an opposite manner and that changes in Θ only play a minor role. The general rule is, decreases in Θ cause idle time to decrease when θ is low and increase when θ is high, given α is high. When α is low, the reverse situation is observed. With respect to backlog, only when α is low and θ is high does a change in Θ have a significant effect. Under these circumstances, a decrease in Θ causes backlog to decrease. Significant

changes in in-process inventory due to changes in Θ occur only when θ is high and are not effected by the value of α. Under the preceding circumstances, a decrease in Θ causes a decrease in in-process inventory. There does not appear to be any significant change in efficiency due to a change in Θ. Significant changes in efficiency result from changes in θ. When viewing the completion date performance, it is noticed that decreases in Θ create improvement when θ is small, and cause a worse situation when θ is large. The value of α is not significant, at least over the range of values tested.

Importance of α. In the preceding discussion, the significance that α plays in the value of the performance criteria is mentioned several times. However, a full analysis of the relative importance of α is presented in this section.

The value of α has no significant effect over the amount of productive time.

With respect to overtime, the level of the overtime value is slightly larger when α is increased, ceteris paribus. However, this difference is not considered significant.

The magnitude of the idle time values increase slightly when α increases, given θ is large. When θ is small, however, an increase in α causes idle time to decrease. When θ is small, this decrease is significant. When θ is large, the decrease is only noticeable.

When considering backlog, there is no significant difference caused by changes in α. Noticeable differences do not conform to a consistent pattern.

The three final effectiveness criteria, in-process inventory, efficiency, and completion date performance, exhibit no noticeable change when changes occur in α, ceteris paribus.

Because of the few performance criteria showing significant changes when α assumes different values, α is considered the parameter of least relative importance. This fact is consistent with the general theory of the capacity requirement algorithm, since α is concerned with the frequency of capacity adjustment, not the magnitude of adjustment. Magnitude being of greater importance than frequency, substantiates the tertiary role played by α.

CHAPTER V

SUMMARY AND CONCLUSIONS

Chapter V is composed of three sections which are intended to round out this research effort. The first section is a brief summary of the problem under consideration. The second section enumerates the conclusions that were reached and the third section suggests areas for future research.

Summary

The purpose of this thesis was to develop and test the effectiveness of a statistically based algorithm which is used to set and adjust basic capacity levels at machine centers in a job shop manufacturing system. Chapter III developed the theory and concept of the algorithm. Chapter IV provided the analysis of the simulation test cases that were used to measure the effectiveness of the algorithm as an aid to job shop management. The algorithm was compared against a decision rule which did not allow the basic capacity at the machine centers to change. The relative importance of the three primary parameters of the algorithm (θ, Θ, and α) was determined. In addition, the effect of changes in each of these parameters on the

seven performance criteria (productive time, overtime, idle time, backlog, in-process inventory, efficiency, and completion date performance) was identified and explained.

Conclusions

The following conclusions were reached as a result of the analysis of the output from the nine simulation test cases:

1. The capacity requirement algorithm will yield both better and worse performance criteria than a constant basic capacity decision rule, depending upon the values selected for the parameters θ, Θ, and α.
2. The relative importance of the parameters θ, Θ, and α is as follows:
   a. The planning period, θ, has the greatest effect upon the performance criteria.
   b. The number of observations of $\bar{x}_{k\theta_\varphi}$ used to establish $\bar{\bar{x}}_{k\theta_\varphi}$, Θ, is of secondary importance.
   c. The confidence level, α, plays a tertiary role in effecting performance criteria.
3. As θ is increased over the range of values tested, improvement is observed in the following performance criteria: productive

time, overtime, backlog, and in-process inventory. Adverse effects are observed in idle time, efficiency, and completion date performance.

4. The effects of changes in Θ on the performance criteria are more subtle than the effects of changes in θ and depend upon the relative values of θ and α. These effects are as follows:

   a. Changes in Θ produce no significant effect on productive time.

   b. Decreases in θ cause increases in overtime, given θ and α are both small. When θ is large, decreases in Θ cause overtime to be reduced considerably regardless of the value of α. When θ is small and α is large, decreases in Θ also cause decreases in overtime, but to a lesser degree.

   c. Decreases in Θ cause idle time to decrease when θ is small and increase when θ is large, given α is large. When α is small, the reverse situation is observed. When θ and α are both small or both large, changes in Θ do not have a significant effect on idle

time.

d. When α is small and θ is large, a decrease in Θ causes backlog to decrease. This is the only situation where changes in Θ cause backlog to change.

e. Significant changes in in-process inventory due to changes in Θ occur only when θ is large and are not affected by the value of α. Under the preceding circumstances, a decrease in Θ causes a decrease in in-process inventory.

f. There does not appear to be any significant change in efficiency due to a change in Θ.

g. Decreases in Θ cause completion date performance to improve when θ is small, regardless of the value of α. When θ is large, the reverse situation is observed.

5. The effect of changes in α on the performance criteria are only minor and are described as follows:

a. Changes in the value of α, over the range of values tested, have no effect on the following performance

criteria: productive time, backlog, in-process inventory, efficiency, and completion date performance, ceteris paribus.

b. Given θ is large, overtime values tend to increase slightly when α is increased, but the magnitude of the change is not significant.

c. The magnitude of the idle time values increases slightly when α increases, given Θ is large. When Θ is small, however, an increase in α causes idle time to decrease. If θ is also small, this decrease is significant. If θ is large, the decrease is only noticeable.

Areas for Future Research

There are several possibilities for additional research in the area of determining capacity requirements at machine centers in a job shop manufacturing system. Some of the more interesting subjects to be addressed are:

1. Determine optimal values for the parameters θ, Θ, and α for given performance criteria.
2. Investigate the importance of the order releasing mechanism and the overtime triggering mechanism.
3. Determine the effect of changing the

assumption on the form of the flow time distribution from a normal to a Poisson or Weibull distribution.

4. Determine the effect that different levels of machine shop loading and different numbers of machine centers have on the performance criteria values.
5. Compare performance criteria under the capacity requirement algorithm to other decision rules besides a constant basic capacity; e.g., changing basic capacity at a machine center by some constant factor, say 1.3, when the z number at that machine center ≤ -2.00.

BIBLIOGRAPHY

(1) Cramer, H. Mathematical Methods of Statistics. Princeton, New Jersey: Princeton University Press, 1954.

(2) Subrahmanyam, V. "The Distribution of Total Flow Time in a Job Shop." (M.S. Report, Oklahoma State University, 1965).

(3) Giffler, B., G. L. Thompson, and V. Van Ness. "Numerical Experience With the Linear and Monte Carlo Algorithms for Solving Production Scheduling Problems." Industrial Scheduling, J. F. Muth and G. L. Thompson. Englewood Cliffs, New Jersey: Prentice Hall, 1963, Chapter 3, pp. 21-38.

(4) Bowman, E. H. "The Schedule-Sequencing Problem." Journal of Operations Research, Vol. 7 (1959), pp. 621-624.

(5) Gomory, R. E. "Outline of an Algorithm for Integer Solutions to Linear Programs." Bulletin of the American Mathematical Society, Vol. 64 (September, 1958).

(6) Dantzig, G. B. "Discrete-Variable Extremum Problems." Journal of Operations Research, Vol. 5 (1957), pp. 266-276.

(7) Markowitz, H. M., and A. S. Manne. "On the Solution of Discrete Programming Problems." Econometrica, Vol. 25, No. 1 (1957).

(8) Andrus, W. E., and A. L. Becker. "A Production Schedule Computed on the IBM 701." IBM Technical Report No. 105,007,273 (January 20, 1954), in Analysis of Industrial Operations, E. H. Bowman and R. B. Fetter, eds. Homewood, Illnois: R. D. Irwin, 1959.

(9) Manne, A. S. "On the Job-Shop Scheduling Problem." Journal of Operations Research, Vol. 8 (1960), pp. 219-223.

(10) Dantzig, G. B. et al. "Upper Bounds, Secondary Constraints, and Block Triangularity in Linear Programming." Econometrica (April, 1955).

(11) Conway, R. W., B. M. Johnson, and W. L. Maxwell. "An Experimental Investigation of Priority Dispatching." Journal of Industrial Engineering, Vol. XI, No. 3 (May-June, 1960), pp. 221-229.

(12) Conway, R. W. "Priority Dispatching and Job Lateness in a Job Shop." Journal of Industrial Engineering, Vol. XVI, No. 4 (July-August, 1965), pp. 228-237.

(13) Rowe, A. J., and J. R. Jackson. "Research Problems in Production Routing and Scheduling." Journal of Industrial Engineering, Vol. VII, No. 3 (May-June, 1956), pp. 116-121.

(14) Boeing Research Task Group. "An Urgency Number System for Job Shop Scheduling." Operations Research Group Report Number OR-64-3 (December, 1964).

(15) Dzielinski, B. P., C. T. Baker, and A. S. Manne. "Simulation Tests of Lot-Size Programming," Industrial Scheduling, J. F. Muth and G. L. Thompson. Englewood Cliffs, New Jersey: Prentice-Hall, 1963, Chapter 9, pp. 127-155.

(16) United States Department of Commerce, National Bureau of Standards. Experimental Statistics Handbook 91. Washington 25, D. C.: U. S. Government Printing Office.

(17) Dunlap, L. L. "The Effect of Order Size on the Operation of a Hypothetical Job Shop Manufacturing System." (Unpub. Ph.D. Dissertation, Oklahoma State University, 1967).

APPENDIX A

JOB SHOP SIMULATOR

```
$JOB            0,15,40000      4640001G CURRY        C3KEC101N
$SETUP 12       DISK,PRINT
$ATEND          00000,77777,3
$EXECUTE        IBJOB
$ID             CURRY-  OSU - NEW - JOB SHOP SIMULATOR - 9-12-65
$IBJOB KEC101   MAP
$IBFTC KEOR1
C      MAIN PROGRAM
C      BOEING JOB SHOP QUEUE SEQUENCING EXPERIMENTAL SIMULATOR
C      GUY CURRY.    WICHITA    SEPT. 65
C      OKLA. STATE UNIV. MODEL
       DIMENSION  ISA(1000,7), SA(1000), ISAD(50)
       DIMENSION  KAP(15,10), KS(11,10), M(3,400)
       COMMON POOL, PNBR, SMDTE, TMDTE
       COMMON NL, NR, NPART, KDATE, LHOLD, L9, L2
       COMMON NCD, NQUE, NRT, NLIST, NPROG, LAST, L3, L4, L5, L6, L7, L8
       COMMON L(20,400), K(200,10), C(24,10), NEW(12)
      0COMMON /LOUT/ ARRIV (5,10), SARRV (5,10), MACHT(5,10), COSTS(5,10)
      1, XBAR(30,10)
 9001 FORMAT (1H1)
 90020FORMAT (4H NEW I4, 7H, MDATE I4, 7H, IDATE I4, 11H, MACH TIME I4,
     111H   ROUTING  10I6)
 90030FORMAT (11H WORK LISTS I4,7H, ZMALL F5.1,7H, ZMFLR F5.1,7H, ZMQUE
     1F5.1,7H, ZMBLG F5.1,6H, ZAVG F5.1, 7H, ZAFLR F5.1,9H, ZACDQUEF5.1,
     29H, ZACDBLG F5.1, / 4X 6HRATE C F5.1,10H, SUM XBAR F5.1,
     38H, RATE M F5.1,12H, MEAN MDATEF5.1,1H,F5.1,1H,F5.1,11H, AVG MDATE
     4F5.1,1H, F5.1,1H,F5.1 / 4X 12HWORK ON HAND F6.1,1H, F6.1,1H, F6.1,
     511H, NBR PARTS I4,1H, I4, 1H, I4,22H (SYSTEM, FLOOR, POOL)
     620H, ALLOW NEW ARRIVALS F7.4)
 90040FORMAT (8H OP CODE I3,5H, CAP I3,2(2H + I3),6H, LOWR I3,6H, XBAR
     1I3,6H, UPPR I3,14H, WORK IN SYST   F5.1,6H, BKLG F5.1,7H, MDATE
     2F6.1,9H, WAIT MN F4.1,4H VAR F4.1 / 4X 5HZMEAN 4(F5.1,1H,),5H ZAVG
     32(F5.1,1H,),10H WORK LIST 5(2I5,1H,))
 9005 FORMAT (5X 10(2I5,1H,))
 9006 FORMAT (8H OP CODE I3, 21H EXHAUSTED WORK LIST.)
 9007 FORMAT (8H OP CODE I3, 15H OUT OF WORK AT I3, 7H TENTHS)
 90080FORMAT (9H PART NO. I4,7H BEFORE I4,7H, MDATE I4, 7H, IDATE I4,
     114H, POOL RELEASE I4,6H, MACH I4,10H TENTHS INI3,14H OPERNS, AVG Z
     2F6.2,11H SINCE POOL)
 90090FORMAT (18H MACHINE TIME USED I4,4H WAS I4,16H OUT OF CAPACITY I4,
     12(2H + I3),16H, WITH IDLE TIME I3,2(2H + I3),31H, S-J EFFIC.  DAY
     2PERIOD CUMUL.)
 90100FORMAT (7X 11HSHOP EFFIC. 3F5.2,11H, OP CODES 4(I3,3F5.2,1H,) /
     16X 5(I3,3F5.2,1H,),I3,3F5.2)
 9011 FORMAT (17H EXPEDITE OP CODE I3, 7H TYPE I I3,9H, TYPE II I3)
 9151 FORMAT (1HO 6HMEAN = F5.1,5X 4HSD = F5.1 / 1H )
 9152 FORMAT (8H   DISTR 2I6)
 91010FORMAT (1H1 10X 9HGUY CURRY 3X 15HBOEING JOB SHOP 3X
     119HCHECK OUT  10-14-65  )
 91030FORMAT (55HOCAPACITY ADJUSTMENT AUG. 1965  PLANNING PERIOD (NXBAR)
     1I3,29H, NBR OF OBSERVATIONS (NXDBL) I3 / 32X 29HCONTROL LIMITS SET
     2 AT (TLOWR) F6.3,12H AND (TUPPR) F6.3,20H STANDARD DEVIATIONS /
     332X 36HEXPECTED ARRIVALS MODIFIED BY (WBAR) F5.2,38H OF QUEUE MACH
     4INE TIME WITH NEGATIVE Z / 32X 28HRELEASE FROM POOL AT (ZSTAR)
     5F6.2 / 32X 35HMAXIMUM INCREASE PER CHANGE (KAPIN) I3,21H, MAXIMUM
```

```
     6CAP (KAPMX) I3,12H TENTHS DAYS)
91050FORMAT (51H EXPEDITE (OVERTIME OR FARMOUT) AT Z-NUMBER (ZOVT1)
     1F6.2,14H UP TO (MOTR1) I4,27H PER CENT OF BASIC CAPACITY / 18X
     233HAND IF NEEDED AT Z-NUMBER (ZOVT2) F6.2,14H UP TO (MOTR2) I4,
     327H PER CENT OF BASIC CAPACITY / 74H Z-NUMBER SEQUENCING IN QUEUES
     4 WITH Z-INTERVALS (SHORTEST FIRST) OF (INTZ) I3,7H TENTHS)
91080FORMAT (45HOWAITING WEIGHTING CONSTANTS -- HISTORIC (W1) F5.2,
     124H, IMMEDIATE BACKLOG (W2) F5.2,21H, WORK IN SYSTEM (W3) F5.2 /
     232X 50HIN REVISING MEAN AND VAR, CURRENT FRACTION (TAPER) F7.4,
     314H UP TO (TAPRM) F7.4,9H EACH DAY /
     421H ORDER GENERATOR L2 = I3,29H (REVIEW, TRANS., ETC.), L3 = I5,
     523H (RATE OF ORDERS), L4 = I4,16H (INITIAL LOAD), / 5X 4HL5 = I5,
     623H (NBR OPERATIONS), L6 = I5, 6H  L7 = I4,19H (SLACK TIME), L8 =
     7I5,6H  L9 = I4,16H (MACHINE TIME), / 5X 18HRANDOM NUMBER (NL) I5,
     85H (NR) I5,20H MULTIPLIED BY 10011)
91110FORMAT (119HOS-J EFFICIENCY BASED ON -- OVERTIME AT 1.5 AND 2. TIM
     1ES BASIC, IN-SHOP SHIFTS AT .25 DAY, EMPLOYEE CHANGES AT 2.0 DAYS/
     290X 2A8,6H   NO. I3)
91210FORMAT (7X 2I3,7X 2F5.3,2(7X I3),6X F4.2,1X 2A8,I3 /
     15(7X F7.4) / 10(4X I4) / 7X I3,2(7X F5.2),2(7X I3),7X F5.2)
91600FORMAT (8H OP CODE I3,10H CUMUL CAP I5,2(1H+ I3),10H, OVERTIME I3,
     116H DAYS, IDLE TIME I4,2(1H+ I3),10H, CAP DOWN I3,9H DAYS SUM I3,
     24H  UP I3,9H DAYS SUM I3)
9153 FORMAT (1HO )
9200 FORMAT(1H1 34HERROR I GREATER THAN NLIST 207 + 2    )
9201 FORMAT (3X 9HM(1,NM) = I5,5X3HNM= I5 )
9202 FORMAT (3X 9HL(1,I) =  I5,5X3HI =  I5  )
9203 FORMAT(12I6)
     READ(5,9203) LAST
100  WRITE(12,9101)
     NADJ = 1
     NRUN = 6
     MCPTY = 10
     RUPPR = 10.
     RLOWR = 4.
     NCD = 10
     NQUE = 200
     NRT = 20
     NLIST = 400
     NPRG  = 400
    0READ(5,9121) NXBAR,NXDBL, TLOWR,TUPPR, KAPIN,KAPMX, WBAR,
    1DATA, DATE, NDATA, TAPER, TAPRM, W1, W2, W3,
    2L2, L3, L4, L5, L6, L7, L8, L9, NL, NR,
    3INTZ, ZOVT1, ZOVT2, MOTR1, MOTR2, ZSTAR
     WRITE(12,9103) NXBAR,NXDBL, TLOWR,TUPPR, WBAR, ZSTAR, KAPIN, KAPMX
     WRITE(12,9105) ZOVT1, MOTR1, ZOVT2, MOTR2, INTZ
     WRITE(12,9108) W1,W2,W3,TAPER,TAPRM, L2,L3,L4,L5,L6,L7,L8,L9,NL,NR
     WRITE(12,9111) DATA,DATE, NDATA
     NPART = 0
     KDATE = 0
     LHOLD = 0
     POOL = 0.
     SMDTE =0.
     TMDTE =0.
     PNBR = 0.
```

```
      IG = 0
      ITOT = 0
      ISQ = 0
      NORRT = NRT - 8
      TAPR2 = 1. - TAPER
      WBAR2 = 1. - WBAR
      RNBAR = 1. / FLOAT(NXBAR)
      RNDBL = 1. / FLOAT(NXDBL)
      RDBLM = 1. / FLOAT(NXDBL - 1)
      X = SQRT(RNDBL)
      TLOWR = X * TLOWR
      TUPPR = X * TUPPR
      ZINT = 10./FLOAT(INTZ)
      DO 101 I=1,50
  101 ISAD(I) = 0
      DO 102 I=1,NLIST
      DO 102 J=1,NRT
  102 L(J,I) = 0
      LTIME = MCPTY * NXBAR
      Y = MCPTY
      DO 109 J=1,NCD
      C(1,J) = 0.
      C(2,J) = 1.5
      C(3,J) = 3.
      DO 103 I=4,24
  103 C(I,J) = 0.
      C(22,J) = LTIME
      DO 104 I=1,3
  104 KAP(I,J) = MCPTY
      KAP(4,J) = 6
      KAP(5,J) = 15
      KAP(12,J) = LTIME
      KAP(13,J) = 0
      KAP(14,J) = 0
      DO 106 I=1,11
  106 KS(12,J) = 10
      DO 107 I=1,NQUE
  107 K(I,J) = 0
      DO 108 I=1,NXBAR
      ARRIV(I,J) = 0.
      SARRV(I,J) = 0.
      MACHT(I,J) = MCPTY
  108 COSTS(I,J) = Y
      DO 109 I=1,NXDBL
  109 XBAR(I,J) = Y
      KSUP = 0
      KSDWN= 0
      MCPTY = MCPTY * NCD
      SUMM = 0.
  301 WRITE(12,9001)
      KDATE = KDATE + 1
      I = 0
      IDATE = KDATE - LHOLD
  392 I = I + 1
      IF(L(1,I) .NE. 0) GO TO 392
```

```
393   IF(I .EQ. NLIST) GO TO 399
      DO 306 J = 1,NORRT
306   NEW(J) = 0
394   NEW(2) = IDATE
      CALL ORDER
      IF(NEW(1) .NE. 0) GO TO 395
      IF(LHOLD .EQ. 0) GO TO 401
      LHOLD = LHOLD - 1
      IDATE = KDATE - LHOLD
      GO TO 394
395   L(1,I) = -NEW(1)
      L(2,I) = NEW(2)
      L(3,I) = IDATE
      DO 396 J = 6,10
396   L(J,I) = 0
      MACH = 0
      DUE = L(2,I)
      DO 307 JJ = 11,NRT
      L(JJ,I) = NEW(JJ - 8)
      IF(L(JJ,I) .EQ. 0) GO TO 307
      L(6,I) = L(6,I) + 1
      LTIME = L(JJ,I) / 100
      CALL INTRN (L(JJ,I), 2, J)
      Y = LTIME
      Y = Y * .1
      C(7,J) = C(7,J) + Y
      C(8,J) = C(8,J) + Y
      X = DUE * Y
      C(10,J) = C(10,J) + X
      C(11,J) = C(11,J) + X
      MACH = MACH + LTIME
307   CONTINUE
      L(5,I) = MACH
      L(7,I) = MACH
      Z = MACH
      Z = Z *.1
      SUMM = SUMM + DUE
      POOL = POOL + Z
      SMDTE = DUE * Z + SMDTE
      TMDTE = DUE + TMDTE
      PNBR = PNBR + 1.
      CALL INTRN (L(11,I), 2, J)
      Y = L(11,I) / 100
      C(9,J) = C(9,J) + Y
      WRITE(12,9002) NEW(1), L(2,I), L(3,I), MACH, (L(J,I),J=11,NRT)
      I = I + 1
      GO TO 393
399   LHOLD = LHOLD + 1
401   DO 402 J = 1,NCD
      IF(C(4,J) .EQ. 0.) GO TO 402
      C1 = TAPER * C(4,J) / FLOAT(KAP(2,J))
      IF(C1 .GT. TAPRM) C1 = TAPRM
      C2 = 1. - C1
      Y = .1 / C(4,J)
      Z = C2 * C(2,J) + C1 * Y * C(5,J)
```

```
      C(3,J) =(C(2,J) * C(2,J) + C(3,J)) * C2 + C1 * .1*C(6,J)*Y - Z*Z
      C(2,J) = Z
  402 CONTINUE
      GO TO 423
  403 DO 404 J = 1,NCD
  404 C(6,J) = C(7,J) * 10. / FLOAT(KAP(2,J))
      ND = KDATE - (KDATE / NADJ) * NADJ
  405 IF(ND .NE. 0) GO TO 421
      DO 406 J = 1,NCD
      KAP(3,J) = KAP(2,J)
      IF(C(6,J) .GE. RUPPR) KAP(3,J) = KAP(2,J) + 2
      IF(C(6,J) .GT. RLOWR) GO TO 406
      IF(KAP(2,J) .LT. 1) GO TO 406
      KAP(3,J) = KAP(2,J) - 1
  406 CONTINUE
  421 IF(ND .EQ. 0) GO TO 423
      C(22,J) = C(22,J) - COSTS(1,J)
      DO 422 I = 2,NXBAR
  422 COSTS(I-1,J) = COSTS(I,J)
      COSTS(NXBAR,J) = KAP(2,J)
      GO TO 501
  423 MCPTY = 0
      N = 0
      NM = 0
      DO 426 J = 1,NCD
      C(22,J) = C(22,J) - COSTS(1,J)
      DO 424 I=2,NXBAR
  424 COSTS(I-1,J) = COSTS(I,J)
      COSTS(NXBAR,J) = KAP(3,J)
      IF(KAP(2,J) .EQ. KAP(3,J)) GO TO 426
      NT = KAP(3,J) - KAP(2,J)
      KAP(2,J) = KAP(3,J)
      IF(NT .GT. 0) GO TO 425
      N = N - NT
      C(12,J) = -NT
      C(13,J) = 0.
      KS(8,J) = KS(8,J) + 1
      KS(10,J) = KS(10,J) - NT
      COSTS(NXBAR,J) = -.125 *FLOAT(NT) + COSTS(NXBAR,J)
      GO TO 426
  425 NM = NM + NT
      C(12,J) = 0.
      C(13,J) = NT
      KS(9,J) = KS(9,J) + 1
      KS(11,J) = KS(11,J) + NT
      COSTS(NXBAR,J) = .125 *FLOAT(NT) + COSTS(NXBAR,J)
  426 MCPTY = MCPTY + KAP(2,J)
      IF(N .EQ. NM) GO TO 501
      IF(N .LT. NM) GO TO 428
      NM = NM - N
      KSDWN = KSDWN + NM
      Z = 1.875 * FLOAT(NM) / FLOAT(N)
      DO 427 J = 1,NCD
  427 COSTS(NXBAR,J) = COSTS(NXBAR,J) + Z * C(12,J)
      GO TO 501
```

```
  428  N = NM - N
       KSUP = KSUP + N
       Z = 1.875 * FLOAT(N) / FLOAT(NM)
       DO 429 J = 1,NCD
  429  COSTS(NXBAR,J) = COSTS(NXBAR,J) + Z * C(13,J)
  501  DO 503 J = 1,NCD
       IF(C(7,J) .NE. 0.) GO TO 5011
       C(5,J) = KDATE
       C(24,J) = 0.
       GO TO 5012
 5011  C(5,J) = C(10,J) / C(7,J)
       Y = C(5,J) - FLOAT(KDATE)
       IF(Y .LT. 1.) Y = 1.
       C(24,J) = C(7,J) / Y
 5012  IF(KAP(2,J) .GT. 0) GO TO 5013
       C(4,J) = C(2,J)
       GO TO 5014
 5013  C(4,J) = W1*C(2,J) + W2*( C(1,J)*10. / (FLOAT(KAP(2,J))+.5)  )
 5014  DO 502 I = 6,11
  502  KAP(I,J) = 0
       DO 503 I = 12,21
  503  C(I,J) = 0.
       I = 0
  504  I = I + 1
       IF(L(1,I) .EQ. 0) GO TO 541
       Y = L(2,I)
       YY = L(2,I) - KDATE
       IF(YY .LT. 1.) YY = 1.
       YY = .1 / YY
       Z = 0.
       CALL INTRN (L(11,I), 2, J)
       IF(L(10,I) .LT. 0) Y = Y + C(4,J)
       DO 505  N = 11,NRT
       IF(L(N,I) .EQ. 0) GO TO 506
       CALL INTRN (L(N,I), 2, JJ)
       C1 = L(N,I) / 100
       Y = Y - C(4,JJ) - C1 * .1 - .45
  505  Z = Z + C(3,JJ)
  506  IF(Z .NE. 0.) GO TO 507
       IF(Y .LT. -.45) Z = -9.8
       GO TO 511
  507  C2 = SQRT(Z)
       C1 = KDATE
       Z = (Y - C1) / C2
       IF(Z .LT. -9.8) Z = -9.8
       IF(Z .GT. 9.9) Z = 9.9
  511  L(9,I) = Z * 1000.
C      IF(L(10,I) .LT. 0) GO TO 513
C      IF(Z .GT. 0) COMPUTE DOWNSTREAM DEMAND NUMBER
       IF(L(1,I) .GT. 0) GO TO 513
       IF(Z .LE. ZSTAR) GO TO 512
       L(4,I) = Y - ZSTAR * C2
       GO TO 513
  512  CALL RELEAS (I,LPLAN)
       L(10,I) = KDATE
```

```
513   IF(L(10,I) .LT. 0) C(20,J) = Z
      LPLAN = L(11,I) / 100
      IF(Z .LE. ZOVT1) KAP(8,J) = KAP(8,J) + LPLAN
      IF(Z .LE. ZOVT2) KAP(9,J) = KAP(9,J) + LPLAN
      IF(Z .LT.0.) KAP(10,J) = KAP(10,J) + LPLAN
      DO 514 N = 11,NRT
      IF(L(N,I) .EQ. 0) GO TO 515
      CALL INTRN (L(N,I), 2, JJ)
      X = L(N,I) / 100
      C(21,JJ) = C(21,JJ) + YY*X
      X = Z * X * .1
      IF ( L(1,I) .GT. 0 ) C(13,JJ) = C(13,JJ) + X
514   C(12,JJ) = C(12,JJ) + X
515   X = Z * .1 * FLOAT(LPLAN)
      C(14,J) = C(14,J) + X
      C(16,J) = C(16,J) + Z
      C(18,J) = C(18,J) + 1.
      IF(L(1,I) .LT. 0) GO TO 521
      C(15,J) = C(15,J) + X
      C(17,J) = C(17,J) + Z
      C(19,J) = C(19,J) + 1.
521   KZ = (Z + 60.) * ZINT + .01
      KZ = KZ * INTZ - 600
      IF(KZ .LT. -98) KZ = -98
      IF(KZ .GT. 99) KZ = 99
      IF(L(1,I) .GT. 0) GO TO 522
      L(8,I) = KZ
      GO TO 523
522   L(8,I) = L(8,I) + KZ
523   IF(L(10,I) .LT. 0) GO TO 504
      IF(Z .LT. 0.) KZ = KZ - 1
      KZ = KZ * 100 + LPLAN
      GO TO 531
530   CALL QUEUE (J, I, KZ)
      GO TO 504
531   IF(K(NQUE - 1, J) .EQ. 0) GO TO 532
      IF(KZ .GE. K(NQUE,J)) GO TO 504
532   NT = NQUE - 3
533   IF(K(NT,J) .EQ. 0) GO TO 534
      IF(KZ .GE. K(NT+1,J)) GO TO 535
      K(NT + 3, J) = K(NT + 1,J)
      K(NT+2,J) = K(NT,J)
534   NT = NT - 2
      IF(NT .GT. 0) GO TO 533
535   K(NT+2,J) = L(1,I)
      K(NT+3,J) = KZ
      GO TO 504
541   NBR = I - 1
      ZMALL = 0.
      ZMFLR = 0.
      ZMQUE = 0.
      ZMBLG = 0.
      ZAQUE = 0.
      ZABLG = 0.
      ZACDO = 0.
```

```
      ZACDB = 0.
      BKLOG = 0.
      BKLGN = 0.
      ALLWK = 0.
      QUEPL = 0.
      DUE   = 0.
      RATEC = 0.
      DO 545 J = 1,NCD
      RATEC = RATEC + C(21,J)
      DO 542 N = 1,NXBAR
  542 ARRIV(N,J) = TAPER * SARRV(N,J) + TAPR2 * ARRIV(N,J)
      DUE   = DUE   + C(10,J)
      ALLWK = ALLWK + C(7,J)
      QUEPL = QUEPL + C(9,J)
      BKLOG = BKLOG + C(1,J)
      BKLGN = BKLGN + C(19,J)
      ZMALL = ZMALL + C(12,J)
      IF(C(7,J) .NE. 0.) C(12,J) = C(12,J) / C(7,J)
      ZMFLR = ZMFLR + C(13,J)
      Y = C(7,J) - C(8,J)
      IF(Y .NE. 0.) C(13,J) = C(13,J) / Y
      ZMQUE = ZMQUE + C(14,J)
      Y = C(1,J) + C(9,J)
      IF(Y .NE. 0.) C(14,J) = C(14,J) / Y
      ZMBLG = ZMBLG + C(15,J)
      IF(C(1,J) .NE. 0.) C(15,J) = C(15,J) / C(1,J)
      ZAQUE = ZAQUE + C(16,J)
      IF(C(18,J) .NE. 0.) C(16,J) = C(16,J) / C(18,J)
      ZABLG = ZABLG + C(17,J)
      IF(C(19,J) .NE. 0.) C(17,J) = C(17,J) / C(19,J)
      ZACDQ = ZACDQ + C(16,J)
  545 ZACDB = ZACDB + C(17,J)
      FLRWK = ALLWK - POOL
      IF(FLRWK .GT. 0.)GO TO 546
      FLRM = KDATE
      GO TO 547
  546 X = DUE - SMDTE
      FLRM = X / FLRWK
      ZMFLR = ZMFLR / FLRWK
  547 IF(ALLWK .GT. 0.) GO TO 548
      DUE = KDATE
      GO TO 549
  548 DUE = DUE / ALLWK
      ZMALL = ZMALL / ALLWK
  549 X = KDATE
      Y = DUE - X
      IF(Y .LT. 1.) Y = 1.
      RATEM = ALLWK / Y
      Y = QUEPL + BKLOG
      IF(Y .NE. 0.) ZMQUE = ZMQUE / Y
      IF(BKLOG .NE. 0.) ZMBLG = ZMBLG / BKLOG
      Y = PNBR + BKLGN
      IF(Y .NE. 0.) GO TO 550
      AVGM = KDATE
      GO TO 551
```

```
550   AVGM = SUMM / Y
      ZAQUE = ZAQUE / Y
551   IF(BKLGN .NE. 0.) GO TO 552
      AVFLM = KDATE
      GO TO 553
552   AVFLM = (SUMM - TMDTE) / BKLGN
      ZABLG = ZABLG / BKLGN
553   ZACDQ = ZACDQ / FLOAT(NCD)
      ZACDB = ZACDB / FLOAT(NCD)
      IF(PNBR .NE. 0.) GO TO 554
      C1 = KDATE
      C2 = KDATE
      GO TO 601
554   C1 = SMDTE / POOL
      C2 = TMDTE / PNBR
601   DO 605 J = 1,NCD
      KAP(1,J) = KAP(2,J)
      IF(KAP(8,J) .EQ. 0) GO TO 605
      IF(K(2,J) .GT. -9900) GO TO 602
      NT = K(2,J) + 10000
      IF(C(20,J) .GT. ZOVT1) KAP(8,J) = KAP(8,J) + NT
      IF(KAP(9,J) .EQ. 0) GO TO 602
      IF(C(20,J) .GT. ZOVT2) KAP(9,J) = KAP(9,J) +NT
602   IF(KAP(8,J) .LE. KAP(1,J)) GO TO 605
      NT = (MOTR1 * KAP(2,J)) / 100
      KAP(1,J) = KAP(1,J) + NT
      IF(KAP(8,J) .GT. KAP(1,J)) GO TO 603
      KAP(1,J) = KAP(8,J)
      KAP(6,J) = KAP(8,J) - KAP(2,J)
      GO TO 605
603   KAP(6,J) = NT
      IF(KAP(9,J) .LE. KAP(1,J)) GO TO 605
      NT = (MOTR2 * KAP(2,J)) / 100
      KAP(7,J) = KAP(9,J) - KAP(1,J)
      IF(KAP(7,J) .GT. NT) GO TO 604
      KAP(1,J) = KAP(9,J)
      GO TO 605
604   KAP(7,J) = NT
      KAP(1,J) = KAP(1,J) + NT
605   CONTINUE
      K2 = KDATE + NXBAR
      DO 617 I = 1,NLIST
      IF(L(1,I) .EQ. 0) GO TO 618
      N = 11
      IF(L(1,I) .GT. 0) GO TO 612
      KTIME = K2 - L(4,I)
      IF(KTIME .LT. 0) GO TO 617
      GO TO 616
612   IF(L(12,I) .EQ. 0) GO TO 617
      KTIME = NXBAR
      LPLAN = L(11,I) / 100
      IF(L(10,I) .GT. 0) GO TO 613
      NT = 0
      GO TO 615
613   CALL INTRN (L(11,I), 2, J)
```

```
  614  NT = C(4,J) * 10. + .5
  615  NT = (NT + LPLAN + 9) / 10
       KTIME = KTIME - NT
       IF(KTIME .LT. 0) GO TO 617
       N = N + 1
  616  CALL INTRN (L(N,I), 2, J)
       LPLAN = L(N,I) / 100
       KAP(11,J) = KAP(11,J) + LPLAN
       IF(N .EQ. NRT) GO TO 617
       IF(L(N+1,I) .NE. 0) GO TO 614
  617  CONTINUE
  618  I = I - 1
       SXBAR = 0.
       EXPAR = 0.
       DO 6201 J = 1,NCD
       Y = 0.
       DO 619 N = 1,NXBAR
       Z = NXBAR - N + 1
  619  Y = Z * ARRIV(N,J) + Y
C      SARRV(1,J) = Y * .1
       EXPAR = EXPAR + Y
       X = (FLOAT(KAP(11,J)) + Y) * RNBAR
       DO 620 N = 2,NXDBL
  620  XBAR(N-1,J) = XBAR(N,J)
       XBAR(NXDBL,J) = X
  6201 SXBAR = SXBAR + X
       SXBAR = SXBAR * .1
       N = PNBR
       NC = I - N
      OWRITE(12,9003) KDATE, ZMALL, ZMFLR, ZMQUE, ZMBLG, ZAQUE, ZABLG,
      1ZACDQ, ZACDB, RATEC, SXBAR, RATEM, DUE, FLRM, C1, AVGM, AVFLM, C2,
      2ALLWK, FLRWK, POOL, I, NC, N, EXPAR
       DO 628 J = 1,NCD
       NC = KAP(14,J)
       C1 = WBAR * FLOAT(KAP(10,J))
       KX = WBAR2 * XBAR(NXDBL,J)  + C1  + .5
      OWRITE(12,9004) J,KAP(2,J),KAP(6,J),KAP(7,J), KAP(4,J),KX,KAP(5,J),
      1C(7,J), C(1,J), C(5,J), C(2,J), C(3,J), (C(I,J),I=12,17),
      2(K(I,J),I=1,10)
       IF(K(11,J) .EQ. 0) GO TO 6203
       IF(K(23,J) .NE. 0) GO TO 6202
       WRITE(12,9005) (K(I,J),I=11,22)
       GO TO 6203
  6202 WRITE(12,9005) (K(I,J),I=11,30)
  6203 C(4,J) = 0.
       C(5,J) = 0.
       C(6,J) = 0.
       IF(KX .LE. KAP(4,J)) GO TO 624
       IF(KX .GE. KAP(5,J)) GO TO 624
       IF(KX .EQ. KAP(2,J)) GO TO 627
       IF(KX .GT. KAP(2,J)) GO TO 622
       IF(NC .LT. 0) GO TO 621
       NC = -1
       GO TO 628
  621  NC = NC - 1
```

```
      IF(NC .GT. -NRUN) GO TO 628
      GO TO 624
  622 IF(NC .GT. 0) GO TO 623
      NC =1
      GO TO 628
  623 NC = NC + 1
      IF(NC .LT. NRUN) GO TO 628
  624 Y = 0.
      Z = 0.
      DO 625 N = 1,NXDBL
      Y = Y + XBAR(N,J)
  625 Z = Z + XBAR(N,J) * XBAR(N,J)
      X = Y * RNDBL
      Z = SQRT((Z - X*Y) * RDBLM)
      KAP(3,J) = WBAR2 * X  + C1  + .5
      NT = KAP(2,J) + KAPIN
      IF(KAP(3,J) .GT. NT) KAP(3,J) = NT
      IF(KAP(3,J) .GT. KAPMX) KAP(3,J) = KAPMX
      NT = TLOWR * Z
      IF(NT .LE. 0) NT = 1
      KAP(4,J) = KAP(3,J) - NT
      IF(KAP(4,J) .GT. 0) GO TO 626
      IF(KAP(3,J) .GT. 1) KAP(4,J) = 1
      IF(KAP(3,J) .EQ. 1) KAP(4,J) = 0
  626 NT = TUPPR * Z
      IF(NT .LE. 0) NT = 1
      KAP(5,J) = KAP(3,J) + NT
  627 NC = 0
  628 KAP(14,J) = NC
      IF(KDATE .GT. LAST) GO TO 151
      DO 633 J = 1,NCO
      IF(KAP(1,J) .GT. KAP(2,J)) WRITE(12,9011) J, KAP(6,J), KAP(7,J)
  633 CONTINUE
  801 NM = 0
      MACH = 0
      IDLE0 = 0
      IDLE1 = 0
      IDLE2 = 0
      COST = 0.
      DO 817 J = 1,NCO
      KTIME = 0
      I = 0
  802 I = I + 1
      IF(I.LT.NQUE) GO TO 8071
  807 WRITE(12,9006)J
      GO TO 804
 8071 N = K(I,J)
      IF(N.NE.0) GO TO 8081
  808 WRITE(12,9007) J,KTIME
      GO TO 804
 8081 NM = NM + 1
      IF(NM.LE.NPRG) GO TO 803
      WRITE(12,9006) NPRG
      GO TO 165
C803  EFFICIENCY, TODAY ON THIS PART, KEFF = PLAN / ACTUAL MACHINE TIME
```

```
  803  M(1,NM) = N
       I = I + 1
       CALL INTRN (K(I,J),2, NT)
       IF(NT.LT.0) NT = NT + 100
       LTIME = KAP(1,J) - KTIME
C      LTIME = ((KAP(1,J) - KTIME) * KEFF) / 100
       IF(NT .LE. LTIME) GO TO 8061
  806  M(2,NM) = J + 1000
       LTIME = LTIME + KTIME
       M(3,NM) = LTIME * 100 + KTIME
       KTIME = LTIME
       GO TO 804
 8061  M(2,NM) = J
       LTIME = KTIME + NT
       M(3,NM) = LTIME * 100 + KTIME
       IF(M(3,NM) .GT. 9999) GO TO 165
C      KTIME = NT / KEFF + KTIME
       KTIME = KTIME + NT
       IF(KTIME .LT. KAP(1,J)) GO TO 802
  804  MACH = MACH + KTIME
       KAP(12,J) = KAP(12,J) - MACHT(1,J)
       DO 8041 I = 2,NXBAR
 8041  MACHT(I-1,J) = MACHT(I,J)
       MACHT(NXBAR,J) = KTIME
       KAP(12,J) = KAP(12,J) + KTIME
       KAP(13,J) = KAP(13,J) + KTIME
       X = COSTS(NXBAR,J) + 1.5*FLOAT(KAP(6,J)) + 2.*FLOAT(KAP(7,J))
       COST = COST + X
       COSTS(NXBAR,J) = X
       C(22,J) = C(22,J) + X
       C(23,J) = C(23,J) + X
       C(12,J) = FLOAT(KTIME) / X
       C(13,J) = FLOAT(KAP(12,J)) / C(22,J)
       C(14,J) = FLOAT(KAP(13,J)) / C(23,J)
       IF(KTIME .EQ. KAP(1,J)) GO TO 816
       KTIME = KTIME - KAP(2,J)
       IF(KTIME .LT. 0) GO TO 814
       KTIME = KTIME - KAP(6,J)
       IF(KTIME .LT. 0) GO TO 813
       KTIME = KTIME - KAP(7,J)
       KS(6,J) = KS(6,J) - KTIME
       IDLE2 = IDLE2 - KTIME
       GO TO 816
  813  KS(4,J) = KS(4,J) - KTIME
       IDLE1 = IDLE1 - KTIME
       GO TO 815
  814  KS(2,J) = KS(2,J) - KTIME
       IDLE0 = IDLE0 - KTIME
       IF(KAP(6,J) .EQ. 0) GO TO 816
       KS(4,J) = KS(4,J) + KAP(6,J)
       IDLE1 = IDLE1 + KAP(6,J)
  815  KS(6,J) = KS(6,J) + KAP(7,J)
       IDLE2 = IDLE2 + KAP(7,J)
  816  KS(1,J) = KS(1,J) + KAP(2,J)
       KS(3,J) = KS(3,J) + KAP(6,J)
```

```
 817  KS(5,J) = KS(5,J) + KAP(7,J)
      IF(NM .NE. NPRG) GO TO 805
      NM = 0
 805  M(1,NM+1) = 0
C     M(2,NM+1) = 0
      KTIME = 0
      LTIME = 0
      N = 0
      Y = 0.
      NT = 0
      Z = 0.
      DO 811 J = 1,NCD
      KTIME = KTIME + KAP(6,J)
      LTIME = LTIME + KAP(7,J)
      N = N + KAP(12,J)
      Y = Y + C(22,J)
      NT = NT + KAP(13,J)
 811  Z = Z + C(23,J)
      WRITE(12,9009) KDATE, MACH, MCPTY, KTIME, LTIME,IDLE0,IDLE1,IDLE2)
      X = FLOAT(MACH) / COST
      Y = FLOAT(N) / Y
      Z = FLOAT(NT) / Z
      WRITE(12,9010) X, Y, Z,  (J, C(12,J), C(13,J), C(14,J),J = 1,NCD)
C     GO TO 202
 2001 DO 201 J = 1,NCD
      DO 2002 I=1,NXBAR
 2002 SARRV(I,J) = 0
      DO 201 I = 1,NQUE
 201  K(I,J) = 0
 202  NEWQ = KDATE + 1
      NM = 0
 203  NM = NM + 1
      IF(M(1,NM).EQ.0) GO TO 301
      CALL INTRN(M(2,NM), 2, J)
 206  I = 0
 207  I = I + 1
      IF(L(1,I).EQ.M(1,NM)) GO TO 210
      IF ( I .GE. NLIST ) GO TO 166
      GO TO 207
 210  CALL INTRR(M(3,NM), 2, MON)
      MACH = (M(3,NM)- MON) / 100
      DONE = MACH
      IF(L(1,I).GT.0) GO TO 2101
      CALL RELEAS (I,LPLAN)
      GO TO 211
 2101 LPLAN = L(11,I) / 100
      IF(L(10,I).LT.0) GO TO 212
      Z = LPLAN
      C(4,J) = C(4,J) + Z
      IQUE = (KDATE - L(10,I))* 10 + MON / 100
C     IF(IQUE.GT.9999) GO TO 165
      QUE = IQUE
      Z = QUE * Z
      C(5,J) = C(5,J) + Z
      C(6,J) = QUE * Z + C(6,J)
```

```
 211  L(10,I) = -MACH
C     IF(MACH.EQ.0) L(10,I) = -1
      GO TO 213
 212  L(10,I) = L(10,I) - MACH
 213  DONE = DONE *.1
      C(1,J) = C(1,J) - DONE
      C(7,J) = C(7,J) - DONE
      C(10,J) = C(10,J) - DONE * FLOAT(L(2,I))
      L(7,I) = L(7,I) - MACH
      IF(M(2,NM).LT.100) GO TO 216
      L(11,I) = L(11,I) - MACH * 100
      K(1,J) = L(1,I)
      K(2,J) = -10000 + L(11,I) / 100
      GO TO 203
C216  CALL MOVE (J)
 216  IF(L(12,I).EQ.0) GO TO 220
      L(10,I) = NEWQ
      DO 217 J = 12,NRT
 217  L(J-1,I) = L(J,I)
      L(NRT,I) = 0
      CALL INTRN (L(11,I), 2, J)
      Z = L(11,I) / 100
      N = NEWQ - L(3,I)+1
      IF(N.GT.NXBAR) GO TO 218
      SARRV(N,J) = SARRV(N,J) + Z
 218  C(1,J) = C(1,J) + Z / 10.
      GO TO 203
 220  Z = .1 * FLOAT(L(8,I))/ FLOAT(NEWQ - L(4,I))
      WRITE (12,9008) L(1,I), NEWQ, (L(J,I),J=2,6), Z
      DUE = L(2,I)
      SUMM = SUMM - DUE
      IF (IG.GE.1000) GO TO 221
      IG = IG + 1
      ISA(IG,1) = L(1,I)
      ISA(IG,2) = NEWQ
      DO 2201 J = 3,7
 2201 ISA(IG,J) = L(J-1,I)
      SA (IG) = Z
      NT = L(2,I) - NEWQ
      NT1 = NT + 21
      IF(NT1.LE.0) NT1 = 1
      IF(NT1.GT.50) NT1 = 50
      ISAD(NT1) = ISAD (NT1) + 1
      ITOT = ITOT + NT
      ISQ = ISQ + NT * NT
 221  IF(L(1,I+1).EQ.0) GO TO 223
      DO 222 J = 1,NRT
 222  L(J,I) = L(J,I+1)
      I = I + 1
      GO TO 221
 223  L(1,I) = 0
      GO TO 203
 158  LINE = 50
 151  WRITE (12,9153)
      DO 155 J = 1,NCD
```

```
 155 OWRITE (12,9160) J, KS(1,J), KS(3,J), KS(5,J), KS(7,J), KS(2,J),
     1KS(4,J), KS(6,J), KS(8,J), KS(10,J), KS(9,J), KS(11,J)
      DO 160 I = 1,IG
      IF(LINE .NE. 50) GO TO 159
      LINE = 0
      WRITE (12,9001)
 159  WRITE (12,9008) (ISA(I,J),J=1,7), SA(I)
      LINE = LINE + 1
 160  CONTINUE
      AIG = IG
      AVG = FLOAT(ITOT) / AIG
      AISQ = ISQ
      SD = SQRT (ABS((AISQ - AIG*AVG*AVG) / (AIG - 1.0)))
      WRITE(12,9151) AVG, SD
      NT1 = -21
      DO 164 I = 1,50
      NT1 = NT1 + 1
 164  WRITE(12,9152) NT1, ISAD(I)
      IF(NDATA .GT. 1) GO TO 100
 165  CALL REMOVE (12)
      STOP
  166 WRITE(12,9200)
      WRITE(12,9201) M(1,NM) , NM
      DO 167 I = 1,NLIST
      IF( L(1,I) .EQ. 0 ) GO TO 158
  167 WRITE(12,9202) L(1,I),I
      GO TO 158
      END
$IBFTC KECS01
      SUBROUTINE RELEAS (I,LPLAN )
      COMMON POOL, PNBR, SMDTE, TMDTE
      COMMON NL, NR, NPART, KDATE, LHOLD, L9, L2
      COMMON NCD, NQUE, NRT, NLIST, NPROG, LAST, L3, L4, L5, L6, L7, L8
      COMMON L(20,400), K(200,10), C(24,10), NEW(12)
     0COMMON /LOUT/ ARRIV (5,10), SARRV (5,10), MACHT(5,10), COSTS(5,10)
     1, XBAR(30,10)
      LPLAN = L(11,I) / 100
      Z = LPLAN
      CALL INTRN (L(11,I), 2, J)
      N = KDATE - L(3,I) + 1
      IF(N .GT. NXBAR) GO TO 2311
      SARRV(N,J) = SARRV(N,J) + Z
 2311 Z = .1 * Z
      C(1,J) = C(1,J) + Z
      C(9,J) = C(9,J) - Z
      Y = L(2,I)
      DO 231 N = 11,NRT
      CALL INTRN (L(N,I), 2, JJ)
      X  = L(N,I) / 100
      X = .1 * X
      C(8,JJ) = C(8,JJ) - X
 231  C(11,JJ) = C(11,JJ) - Y*X
      Z = L(5,I)
      Z = Z / 10.
      POOL = POOL - Z
```

```
      SMDTE = SMDTE - Z*Y
      TMDTE = TMDTE - Y
      PNBR = PNBR - 1.
      L(1,I) = -L(1,I)
      L(4,I) = KDATE
      IF(KDATE .EQ. NEWQ) L(8,I) = 0
      RETURN
 2321 NT = -L(1,I)
      DO 232 N = 1,NQUE,2
      IF(NT .EQ. K(N,J)) GO TO 233
  232 CONTINUE
      RETURN
  233 IF(N+1 .EQ. NQUE) GO TO 235
      NT = N + 2
      DO 234 N=NT,NQUE
  234 K(N-2,J) = K(N,J)
  235 K(NQUE-1,J) = 0
      K(NQUE,J) = 0
      RETURN
      END
$IBFTC KECS02
      SUBROUTINE INTRN (N, I, K)
      IF(I .GT. 3) I = 3
      INO = 10 ** (4-I)
      KA = N / INO
      K = N - KA*INO
      RETURN
      END
$IBFTC KECS03
      SUBROUTINE INTRR (N, I, K)
      N1 = N *  10**I
      KA = N1 / 10000
      K = N1 -  KA * 10000
      RETURN
      END
$IBFTC KECS04
      SUBROUTINE ORDER
      COMMON POOL, PNBR, SMDTE, TMDTE
      COMMON NL, NR, NPART, KDATE, LHOLD, L9, L2
      COMMON NCD, NQUE, NRT, NLIST, NPROG, LAST, L3, L4, L5, L6, L7, L8
      COMMON L(20,400), K(200,10), C(24,10), NEW(12)
     OCOMMON /LOUT/ ARRIV (5,10), SARRV (5,10), MACHT(5,10), COSTS(5,10)
     1, XBAR(30,10)
  701 CALL INTRR(NL, 0, NL)
      CALL INTRR(NR, 0, NR)
      CALL INTRN(NR, 1, KKB)
      NRR = NR / 10 + KKB
      CALL INTRR(NRR, 0, NRR)
      NL = NL * 11 + NR / 1000 + NRR / 1000 + NR
      CALL INTRR(NL, 0, NL)
      CALL INTRR(NRR, 1, NR)
      NR = NR + 3
      IF(NR .GE. L3) GO TO 702
      IF(NPART .LE. L4) GO TO 702
C     NEW(1) = 0
```

```
      RETURN
  702 NPART = NPART + 1
      NEW(1) = NPART
      N = NL / L5 + 3
      CALL INTRN(NL, 2, KKB)
      CALL INTRR(NEW(2), 1, KK1)
      NEW(2) = L6 / (KKB + L7) + KK1 + N * L2
      DO 703 NI = 3,N
      CALL INTRN(NR, 1, KKB)
      NRR = NR / 10 + KKB
      NL = NL * 11 + NR / 1000 + NRR / 1000 + NR
      CALL INTRR(NL, 0, NL)
      CALL INTRR(NRR, 1, NR)
      NR = NR + 3
      CALL INTRN(NL, 2, NRR)
      NRR = L8 / (NRR + L9)
      NEW(2) = NEW(2) + NRR
      CALL INTRR(NRR, 2, KKB)
      NEW(NI) = KKB + NL / 1000 + 1
      CALL INTRR (NEW(NI), 0, NEW(NI))
  703 CONTINUE
      CALL INTRR(NEW(2), 0, NEW(2))
      NEW(2) = (NEW(2) + 5) / 10
      CALL INTRR(NEW(2), 0, NEW(2))
      RETURN
      END
$IBFTC KECS05
      SUBROUTINE MOVE (J)
      COMMON POOL, PNBR, SMDTE, TMDTE
      COMMON NL, NR, NPART, KDATE, LHOLD, L9, L2
      COMMON NCD, NQUE, NRT, NLIST, NPROG, LAST, L3, L4, L5, L6, L7, L8
      COMMON L(20,400), K(200,10), C(24,10), NEW(12)
     0COMMON /LOUT/ ARRIV (5,10), SARRV (5,10), MACHT(5,10), COSTS(5,10)
     1, XBAR(30,10)
      DO 242 JJ = 3,NQUE
      K(JJ-2,J) = K(JJ,J)
  242 IF(K(JJ-1,J) .EQ. 0) RETURN
      K(NQUE-1,J) = 0
      K(NQUE,J) = 0
      RETURN
      END
$IBFTC KECS06
      SUBROUTINE QUEUE (J, I, KZ)
      COMMON POOL, PNBR, SMDTE, TMDTE
      COMMON NL, NR, NPART, KDATE, LHOLD, L9, L2
      COMMON NCD, NQUE, NRT, NLIST, NPROG, LAST, L3, L4, L5, L6, L7, L8
      COMMON L(20,400), K(200,10), C(24,10), NEW(12)
     0COMMON /LOUT/ ARRIV (5,10), SARRV (5,10), MACHT(5,10), COSTS(5,10)
     1, XBAR(30,10)
 9006 FORMAT(8H OP CODE, I3, 21H EXHAUSTED WORK LIST.)
      LPART   = L(1,I)
      IF(LPART .GT. 0) GO TO 553
      DO 552 N=1,NQUE,2
      IF (LPART .EQ. K(N,J)) GO TO 562
      IF (K(N,J) .EQ. 0) GO TO 561
```

```
552   CONTINUE
      RETURN
553   DO 554 N=1,NQUE,2
      IF (LPART .EQ. K(N,J)) GO TO 562
      IF (K(N,J) .LT. 0) GO TO 558
      IF (K(N,J) .EQ. 0) GO TO 561
554   CONTINUE
      WRITE(12,9006) J
      WRITE(12,9006) J
      WRITE(12,9006) J
      RETURN
558   IF (N+1 .EQ. NQUE) GO TO 561
      NT = NQUE - 2
      N1 = NT + N
      N2 = NQUE + N
      DO 559 JJ = N,NT
      KN1 = N1 - JJ
      KN2 = N2 - JJ
559   K(KN2, J) = K(KN1,J)
561   K(N,J) = LPART
562   K(N+1,J) = KZ
      RETURN
      END
```

VITA

Stanley Gordon Jones

Candidate for the Degree of

Doctor of Philosophy

Thesis: ON DETERMINING CAPACITY LEVELS AT MACHINE CENTERS IN A JOB SHOP MANUFACTURING SYSTEM

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Ponca City, Oklahoma, February 8, 1939, the son of Russell O. and Cynthia Marie Jones.

Education: Attended Lincoln Elementary School in Ponca City, Oklahoma; graduated from Ponca City High School, Ponca City, Oklahoma, in May 1957; received the Bachelor of Science degree in Industrial Engineering and Management from Oklahoma State University, Stillwater, Oklahoma, in May 1962; received the Master of Science degree in Industrial Engineering and Management from Oklahoma State University, Stillwater, Oklahoma, in May 1963; completed requirements for the Doctor of Philosophy degree in May 1968.

Professional Experience: Performed methods improvement and motion and time studies as an Engineer Trainee for Dowell, Inc., Alice, Texas; performed plant studies involving statistical quality control, plant layout, equipment utilization, and critical path scheduling as a Junior Engineer with Procter and Gamble, Dallas, Texas; prepared economic evaluations and pipeline design specifications as an Associate Engineer with Continental Oil Company, Houston, Texas; participated in computer simulation studies and model building as an Engineer with Dupont in Wilmington, Delaware; Instructor in the School of Industrial Engineering and Management at Oklahoma State University, Stillwater, Oklahoma; Special

Assistant to the Superintendent of Industrial Engineering in charge of operations research applications to plant problems with The Ethyl Corporation, Baton Rouge, Louisiana; served a two-year active duty tour with the U. S. Army Data Support Command, Pentagon, Washington, D.C., as an ADP Project Officer and Assistant Chief of the Quality Assurance Office, grade of Captain; Assistant Professor in the Department of Systems Management, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.