SYSTEM DESIGN OF AN EVALUATION AID FOR
JOBSHOP SCHEDULING HEURISTICS,

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Rezvan Rashidianfar
June, 1986.
TO MY HUSBAND.....

..... HE KNOWS WHY
ACKNOWLEDGMENT

I wish to express my appreciation to my thesis advisor, Professor Ralph V. Rogers for his devoted attention, enthusiasm, knowledge, and for his helpful encouragement through this work.

I, also, wish to thank Dr. Charles M. Overby and Dr. Samuel Jasper for their support and effort serving on the thesis committee.

Finally, I would like to express my gratitude to my parents Mr. Ali Asghar Bohlool (آقابه عیسی بھلول) and Mrs. Sedigheh Kasray (سیدیه کسرای) for their continuous support and patient during course of my studies.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>3</td>
</tr>
<tr>
<td>A.) The Jobshop Problem</td>
<td>3</td>
</tr>
<tr>
<td>B.) Complexity Theory</td>
<td>4</td>
</tr>
<tr>
<td>C.) Heuristic</td>
<td>6</td>
</tr>
<tr>
<td>D.) Heuristic Evaluation</td>
<td>8</td>
</tr>
<tr>
<td>II. DEVELOPMENT OF HEURISTICS EVALUATION AID</td>
<td>12</td>
</tr>
<tr>
<td>A.) Goals For The Package</td>
<td>12</td>
</tr>
<tr>
<td>B.) Design of The System</td>
<td>12</td>
</tr>
<tr>
<td>1. Jobshop Modeling</td>
<td>14</td>
</tr>
<tr>
<td>2. Jobshop Scheduling</td>
<td>16</td>
</tr>
<tr>
<td>3. Selection of Heuristics</td>
<td>18</td>
</tr>
<tr>
<td>4. Experimental Design</td>
<td>19</td>
</tr>
<tr>
<td>5. Analysis of Data</td>
<td>21</td>
</tr>
<tr>
<td>III. EXAMPLE PROBLEM</td>
<td>24</td>
</tr>
<tr>
<td>A.) Performance Measure; Cmax</td>
<td>27</td>
</tr>
<tr>
<td>B.) Factor 1: Heuristics</td>
<td>27</td>
</tr>
<tr>
<td>C.) Factor 2: Job/Machine Ratio</td>
<td>30</td>
</tr>
<tr>
<td>D.) Factor 3: Lot Size</td>
<td>31</td>
</tr>
<tr>
<td>E.) Lot-Ratio Interaction</td>
<td>37</td>
</tr>
<tr>
<td>F.) Lot-Heuristic Interaction</td>
<td>37</td>
</tr>
<tr>
<td>G.) Ratio-Heuristic Interaction</td>
<td>41</td>
</tr>
</tbody>
</table>
H.) Lot-Ratio-Heuristic Interaction ..... 41

IV. CONCLUSION .................................................. 54

BIBLIOGRAPHY .................................................. 56

APPENDIX 

A. User's Manual .............................................. 60
B. Documentation ............................................. 66
C. Program Flowcharts ................................. 81
E. Program Listing ...................... 106
<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Shop Parameters</td>
<td>25</td>
</tr>
<tr>
<td>2. Modeled Jobshop</td>
<td>26</td>
</tr>
<tr>
<td>3. Jobshop Schedule</td>
<td>28</td>
</tr>
<tr>
<td>4. ANOVA</td>
<td>34</td>
</tr>
<tr>
<td>5. Duncan's Multiple Range Test For Cmax Ratio</td>
<td>35</td>
</tr>
<tr>
<td>6. Duncan's Multiple Range Test For Cmax Lot Size</td>
<td>36</td>
</tr>
<tr>
<td>7. Duncan's Multiple Range Test For Cmax Heuristic</td>
<td>36</td>
</tr>
<tr>
<td>8. Analysis of Variance Lot-Ratio Interaction</td>
<td>38</td>
</tr>
<tr>
<td>9. Analysis of Variance Lot-Heuristic Interaction</td>
<td>38</td>
</tr>
<tr>
<td>10. Analysis of Variance Ratio-Heuristic Interaction</td>
<td>42</td>
</tr>
<tr>
<td>11. Analysis of Heuristic Lot-Ratio-Heuristic Interaction</td>
<td>44</td>
</tr>
<tr>
<td>FIGURE</td>
<td>FIGURE</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1. Lot-ratio Interaction</td>
<td>39</td>
</tr>
<tr>
<td>2. Lot-heuristic Interaction</td>
<td>40</td>
</tr>
<tr>
<td>3. Ratio-heuristic Interaction</td>
<td>43</td>
</tr>
<tr>
<td>4. Ratio-lot-heuristic Interaction at (Small Lot) of Lot Size</td>
<td>46</td>
</tr>
<tr>
<td>5. Ratio-lot-heuristic Interaction at (Large Lot) of Lot Size</td>
<td>47</td>
</tr>
<tr>
<td>6. Ratio-lot-heuristic Interaction at (FCFS) Level of Heuristic</td>
<td>48</td>
</tr>
<tr>
<td>7. Ratio-lot-heuristic Interaction at (RND) Level of Heuristic</td>
<td>49</td>
</tr>
<tr>
<td>8. Ratio-lot-heuristic Interaction at (SPT) Level of Heuristic</td>
<td>50</td>
</tr>
<tr>
<td>9. Lot-ratio-heuristic Interaction at First Level of Ratio</td>
<td>51</td>
</tr>
<tr>
<td>10. Lot-ratio-heuristic Interaction at Second Level of Ratio</td>
<td>52</td>
</tr>
</tbody>
</table>
Heuristics are widely used in jobshop scheduling to determine the order in which jobs are to be processed. The research in this area has been directed to designing general application heuristics. This study offers a methodology and a software package to evaluate scheduling heuristics over general operating characteristics of a jobshop using simulation methods and statistical analysis.
CHAPTER I

INTRODUCTION

The jobshop scheduling has been a problem in many industries for decades. In industrial situations there is a need to determine the best order in which jobs should be processed on machines. In most instances, the scheduling procedure is done by manual methods based upon the experience of some expert scheduler. The difficulties associated with this method are many. One obvious problem in manual scheduling is the time it takes to produce a feasible solution, especially when a large, complex process is involved. In such complex operations, more systematic and formalized systems for scheduling are needed. Large and complex situations require more efficient and timely scheduling than can be provided by manual methods.

Extensive research has been done for several decades in attempts to improve on the methods of manual scheduling. As a result, many heuristics have been developed and offered for use [15-26]. Unfortunately, industry has not been able (or willing) to make much use of these developments, because actual scheduling environments are highly variable and difficult to model rigidly. Technological constraints and product mix may change from week-to-week or hour-to-hour. Production schedulers claim
that their scheduling environment is not only unique, but sufficiently different from any other setting to require problem specific solutions [39]. Such claims are even understandable given the discussion of most scheduling approaches in the literature.

The approaches found in the literature treating scheduling theory are typically found to be problem specific, detailing how a particular approach was employed to resolved a problem facing the investigators. While this is not undesirable in itself, it does leave the scheduler and other investigators to interpret the approach in terms of the problem of interest to them with no common bases for evaluation and comparisons.

What is needed is a readily available tool to aid investigators and scheduler in evaluating and comparing scheduling approaches which is general enough to capture a wide variety of jobshop situations, which is relatively easy to use and implement, and which provides the capability and basis for strong statistical analysis. This report addresses this need and offers a computer software package to meet it.
BACKGROUND :

Given the general topics of jobshop scheduling an extensive literature search was performed. The objective of reviewing published research materials was threefold: 1.) to determine the amount of research done on scheduling; 2.) to identify attempts to comparatively evaluate jobshop scheduling heuristics and 3.) to identify those issues necessary to lead to the goal of developing a computer aid for investigators and schedulers to evaluate jobshop scheduling heuristics.

A. THE JOBSHOP PROBLEM:

The general jobshop problem consists of $n$ jobs \( \{ j_1, j_2, \ldots, j_n \} \) to be processed through $m$ machines \( \{ m_1, m_2, \ldots, m_m \} \). Each job must pass through each machine once and only once. The processing of a job on a machine is called an operation. The operation on the $i$th job by the $j$th machine is denoted as $O_{ij}$. Technological constraints demand that each job should be processed through the machines in a particular order. For general jobshop problems there are no restrictions upon the form of the technological constraints. Each job has its own processing order and this may bear no relation to the processing order of any other job. Each operation $O_{ij}$ takes a certain length of time, the processing time, to perform. This is denoted by $P_{ij}$. By convention any
time required to adjust, or set up, the machine to process this job is included in $P_{ij}$. Also, any time required to transport the job to the machine is included. The processing time $P_{ij}$ is fixed and known in advance [6].

The jobshop scheduling activity is obviously vital in any business. The adequacy of a scheduling system has a great effect on costs. This fact has captured the interest of a great many researchers and schedulers as evident by the number of books [4,5,6,7] and survey articles in the literature[8,9]. The attention directed at this problem is stimulated by the substantial gains to be achieved in meeting due dates, the increase in machine utilization, higher productivity, etc. Many algorithms such as implicit enumeration [10,11], branch-and-bound[12], and mixed integer programming[13], have been developed to provide optimum solutions. However, these methods cannot successfully be applied to practical scheduling problems because of the NP-complete characteristic of the problem as identified in complexity theory which makes it infeasible to obtain an optimal solution for all but the simplest problems.

B. COMPLEXITY THEORY:

Complexity theory seeks to classify problems according to the mathematical order of the computational resources required to solve the problem via digital
computer algorithms. For example, if some algorithm requires a number of computations or an amount of computational time that can be expressed as fourth order polynomial of the problem size, \( n \), the algorithm is said to be of the order \( n^4 \), or simply \( O(n^4) \). Other algorithm may be \( O(n^2 \log n) \), \( O(n) \), \( O(2^n) \), etc. The goal is to define the fundamental frontiers separating problems and algorithms into truly distinct classes. Of particular interest are three classes of problem size: (i) computation that grows logarithmically (or slower), (ii) computation that grows polynomially, and (iii) computation that increases exponentially (or faster). Although these three categories may seem rather crude, they provide enough resolution to gain a great deal of fundamental insight into the tractability of a problem[39].

The Class-NP of decision problems is characterized by what may be called an "easy-to-verify-but-not-necessarily-easy-to-solve" property. For example, in the flow-shop case a proposed schedule (i.e. a candidate solution) is provided. It would be very simple and quick to determine whether the schedule met all feasibility requirements and the makespan threshold. If the supplied schedule existed, the problem would be settled. But if the schedule violated some constraint, all that would be known was there would be a need to look for another candidate. Specifically, the Class-NP includes all decision problems that could be
polynomial-time solved if the right (polynomial length) "clue" or "guess" were apprehended to the problem [39].

The terminology NP derives from the fact that NP problems are nondeterministically polynomial-time solvable in the sense that a random guess generator has some positive probability of resolving the problem whenever the correct response is "yes." For discrete optimization, the NP problem may be best conceptualized as problems which can be solved by complete enumeration of some polynomial length decision vector [39].

A problem which is a member of the Class-NP is called NP-Complete if every other problem in NP is polynomial reducible. Thus, NP-Complete forms a subclass of NP. It should be reiterated that the concept of NP-Completeness applies to decision problems. In saying that a scheduling problem is NP-Complete, what is actually being referred to is the decision analog of the optimization problem. Hence, given the reducibility of NP-Complete problems, when a decision problem cannot be solved in polynomial-time, then its analog optimization problem cannot be solved in polynomial time. In such cases, the analog optimization problem is said to be NP-Hard.

C. HEURISTIC

Complexity theory has led to the implication that large, complex jobshop scheduling problems probably cannot
be solved for an optimum schedule in some viable time frame. However, in reality we cannot leave the problem unsolved. Scheduling problems have a realistic basis and exist because the processing of jobs has to be sequenced. Upon this need some schedule is used, and from experience it is known that the cost of processing depends on the choice of schedule. If a final optimal solution for the problem cannot be found, all analysis should not be abandoned, but the knowledge used to find a schedule which may at least be expected to perform better than average. As a result, we prefer heuristic methods to an enumerative one. These constructive solutions or approximation algorithms produce solutions which may be within a certain distance of the optimum[6]. They appear to be the obvious solution to practical jobshop scheduling problems.

In the past two decades researchers in the field of scheduling have developed over 100 priority rules through simulation techniques for constructive solution methods. Panwalker and Iliskader [9] have presented a survey of scheduling rules to be used by practitioners and researchers. In an attempt to classify scheduling rules into several categories, distinction is made between synonymous terms such as priority rules, heuristics, and scheduling rules. Priority rules are considered a technique by which a number is assigned to each waiting job according to some method and the job with minimum value is selected [27].
Heuristic is defined as some "rule of thumb" whereas a scheduling rule can consist of a combination of one or more priority rule or heuristics. Further division is made at each of the three categories. Simple priority rules are broken down into several related areas. Under this category, nine rules are developed related to processing time, two rules related to due date, four rules related to number of operations, ten rules related to arrival time and random, etc. In the same manner, twenty-five other rules are developed under the heuristics title. All in all, a total of 113 rules of all shape exist to be used by the scheduler. However, this is where the problem of which heuristic or priority rule to use arises.

D. HEURISTIC EVALUATION.

The application of heuristics has been the focus of many studies [15-26]. However, few comparative studies have been done on heuristic performance. Dannenbring [28], in an attempt to evaluate heuristics, presented a computational experience with eleven flow shop sequencing heuristics. In addition, he included three previously unreported heuristics, one of which turned out to be superior to the other ten heuristics tested. The comparisons were made on a variety of problem sizes, up to fifty jobs and fifty operations. The processing times for
the problem were randomly generated integers uniformly distributed over the interval (0-99). In testing these heuristics some evaluatory measures were taken, such as: a) relative error measured from the 1.) optimum or estimate of the optimum, 2.) best lower bound, and b) sample quality and consistency. The computational result of this evaluation provided information that for a given flow shop some heuristics performed better than the others. A more important finding of this study was that four of the five worst performing algorithms are single-shot solution generating heuristics, or those that generate a single solution only. This substantiated the intuitive notion that multiple-shot and solution improvement heuristics are preferable to those that consider only one solution.

Another approach in evaluating of scheduling heuristics is developed by Bunnag and Smith[29]. Their method consists of three parts. First, a generalized objective function is formulated which is the sum of costs of tardiness, carrying in-process inventory and machine idleness. Second, a multifactor priority rule is developed, which is a weighted average of four factors used in simple priority rules. Third, a method is presented for using a computer search techniques to determine the best weights to use in the priority rule.

The results obtained from the above two methods have not been statistically supported and in fact very
few have approached comparative studies employing a strong statistical basis. One such study has been conducted by Russell and Taylor [30]. Their work was aimed at evaluating scheduling policies in a dual resources constrained assembly shop. The scheduling policies examined included duedate assignment, labor assignment and item sequencing rules. The data for analysis was generated by a SLAM II simulation model of a hypothetical dual constrained assembly shop operation. A complete factorial experimental was employed to determine whether job structure, duedate assignment rule, labor assignment rule and item sequencing rule or their interaction significantly affects the root mean square of tardiness of jobs completed by the assembly shop. Further analysis was made to identify where significant differences in performance occur by using the Tukey multiple range test.

Another more general approach has been under taken by Rochette and Asdowski [31], in which they developed and analyzed a particular set of real world jobshops. The characteristics of the process include assembly operations. An experimental design was used which allowed statistical comparisons to be made. Various control and staffing polices were considered, with mean flow time and mean tardiness used as performance criteria. The shortest imminent operation time rule was found to provide the best
performance in all but one situation. Workforce flexibility was observed to significantly improve both performance criteria for all dispatching rules considered.

In conclusion, the existing evaluation studies as documented above are based on specific examples and problems, and the findings may not necessarily be true for any given jobshop and are not ammerable to readily evaluation of other scheduling studies. Thus, none have offered an easily transferable tool to serve as basis for evaluating scheduling heuristics over general operating characteristics of a jobshop. Thus, the proposed software package provides the means to perform scheduling heuristics evaluation and analysis using a strong statistical basis for use in any given jobshop.

The lack of evidence of a broad experimental design on scheduling heuristics confirm that this is a new approach to evaluation of scheduling procedures. For this reason the study was undertaken.
CHAPTER II

DEVELOPMENT OF HEURISTICS EVALUATION AID

The following discussion provides information concerning the goals and specific requirements for the design of a software package to serve as an evaluation aid for jobshop scheduling researchers and schedulers.

A. GOALS FOR THE PACKAGE

The goal for a jobshop scheduling evaluation aid are: 1) to provide a methodology for evaluation of various jobshop scheduling heuristics over a wide range of jobshop situations; 2) to provide a package flexible enough so that it can easily be adapted to a variety of situations and requirements; 3) to be easily use for user with little previous computer knowledge or programming experience; 4) to carry out the evaluation, incorporating a strong statistical basis for evaluation; 5) and, finally, to provide graphical presentations of the evaluation results for easy interpretation.

B. DESIGN OF THE SYSTEM

The design of this software package is integrated into three parts: jobshop modeling and scheduling, experimental design, and analysis of results. Each part is
designed according to certain requirements which will be discussed in detail later on.

The software to implement jobshop modeling and scheduling is developed using Pascal language. The structured computer language Pascal was chosen for its modularity to best accommodate the flexibility required. This characteristic allows for addition and modification of new heuristics, evaluation factors, and distribution without major modification to other parts of the software package. Thus, it consists of a group of procedures each performing and satisfying a specific requirement. These procedures are linked together through a main driver module which controls the direction of execution.

In order to model the jobshop, the user is asked to provide the shop parameters, number of machines, and processing time distribution. Before this jobshop can be scheduled, heuristics need to be selected so that the jobshop can be scheduled accordingly.

The performance of these heuristics is measured through experimental design. Depending on the criteria decided by the user, a mathematical model representing the experiment is developed. Various statistical tests are conducted on this model to analyze the impact of heuristics on these criteria. The tests are performed by an existing software package, Statistical Analysis System (SAS).
The results obtained from the statistical analysis is in tabular form. To make the interpretation of these results easier, the graphical software package TELAGRAF is used to create visual representations of the results. From these graphs, it can be determined which heuristics performed better for that particular jobshop.

The major topics of this overview will be discussed in detail next.

1.) JOBSHOP MODELING

The first part of this system design is to model a jobshop. Several assumptions are made at this point about the jobshop to be modeled. They are: 1) all jobs have same number of operations; 2.) each job is processed only once on each machine; 3.) the number of jobs is a ratio of the number of machines; 4.) there must be at least as many machines as maximum number of operations; 5.) the processing distribution time of the machines is normally distributed; and 6.) the number of machines and their processing time distribution are provided by the user. However, these assumptions can be modified as the user needs. Through the literature search, it was suggested that the hardest jobshop to schedule is one with the same number of operations for all jobs. But in case the user desires to have variety in the number of operations,
the software package will support it. The third assumption regarding the job/machine ratio was made just for the experimental design purpose. If the user wishes to have a fixed number of jobs and choose to have a different factor, he/she may do so. Although, the processing time distribution is assumed to be normal, other distributions can be incorporated with the addition of new procedures to represent these distributions.

The specific requirements with regard to jobshop generation are: 1.) assign operations to machines; 2.) assign processing time to operations; 3.) assign operation processing order for each job. To meet these requirements and above assumptions, Monti Carlo methods were employed. In particular, the technique used to assign processing time to operations is a scheme employing random numbers, that is $\sim(0,1)$ random variables, which is used for solving deterministic problems[32]. Under this technique, various functional relationships are used in order that certain complicated probability density functions can be represented in a similar form in which deviates can easily be generated by the more common technique [33] (for detailed information on the software to implement these techniques refer to appendix B, documentation of the package). Applying these techniques, the jobshop scheduling situation is generated. The next task is to schedule this jobshop situation.
2. **JOBSHOP SCHEDULING**

The software to implement the scheduling of the jobshop is developed using the Branch-and-bound algorithm which is the principle basis for implementation of most heuristic decision rules in the literature. This algorithm in its most general form generates all the active schedules for a jobshop problem. In an active schedule the processing sequence is such that no operation can be started any earlier without delaying some other operation or violating the technological constraints. This algorithm, which constructs active schedules, was developed by Giffler and Thompson [35]. Before explaining the method, some notation and terminology need to be defined. First, operations should be scheduled one at a time. Second, an operation is schedulable if all those operations which must precede it within its job have already been scheduled. Third, since there are nm operations (m machines and n jobs), the algorithm will repeat through nm stages. The following notations are defined at stage t.

- \( P_t \) - is the partial schedule of the (t-1) scheduled operation.
- \( S_t \) - is the set of operations schedulable at stage \( t \), e.g. all the operations that must precede those in \( S_t \) are in \( P_t \).
- \( \sigma^t_k \) - is the earliest time that operation \( 0 \) in \( S_t \) could be started.
is the earliest time that operation $O_k$ in $S$ could be finished, that is $\phi_k = \phi_k' + p_k$, where $P$ is the processing time of $O_k$.

STEP 1: Let $t = 1$ with $P$ being null. $\phi_k'$ will be the set of all operations with no predecessors; in other word, those that are first in their job.

STEP 2: Find $\phi^* = \min_0 \phi_k$ in $S$ and the machine on which $\phi^*$ occurs. If there is a choice for $M$, choose arbitrarily.

STEP 3: Choose an operation $O_k$ in $S$ such that

1.) it requires $M$, and
2.) $\phi_j < \phi^*$

STEP 4: Move to the next stage by

1.) adding $O_j$ to $P$, so creating $P_{t+1}$
2.) deleting $O_j$ from $S$ and creating $S_{t+1}
$ by adding to $S$ the operation that directly follows $O_j$ in its job (unless $O_j$ completes its $j$ job);

3.) incrementing $t$ by 1.

STEP 5: If there are any operations left unscheduled ($t < n$), go to step 2; otherwise, stop.

Since the set of active schedules is guaranteed to contain an optimum, generating all and selecting the best leads to an optimum solution. Although this process of complete enumeration is feasible for small problems, as the theory of NP-completeness predicts, such
an approach rapidly becomes computationally infeasible as the problem size grows. Therefore, for most problems we cannot avoid the problem of making a choice at step 3 by making all the possible choices. We must be more selective.

It is at step 3 that priority rules or heuristics are employed specifically for this purpose. The following is a more detailed discussion on the selection of heuristics.

3. SELECTION OF HEURISTICS:

As we mentioned above at step 3 of the branch-and-bound algorithm, some decision need to be made with regard to the next job to be scheduled. This decision is based on the heuristic chosen to govern the scheduling at step 3. The limitation and assumption made for the heuristics selection depends on the type of jobshop being scheduled and the requirements associated with it. One of the main reasons for obtaining conflicting results in various simulation experiments involving a given rule is the difference in operating conditions. It is therefore essential to know the type of assumptions and shop parameters used before scheduling heuristics can be selected for proper application. At the time of evaluation, the users provide however many heuristics to be evaluated with their jobshop. The limitation is left to the users to be made with regard to their particular jobshop and heuristics that seem to work best for them.
When heuristics are chosen, they can be coded and easily incorporated into the package.

Since this software package is an aid to evaluate heuristics, it should be able to evaluate a wide variety of heuristic applied to jobshop scheduling without any limitation. As part of the package, several heuristics are included to be used for illustration of the method. A detailed discussion of the software associated with these tasks can be found in appendix B.

4. EXPERIMENTAL DESIGN:

An experiment may be defined as a study in which certain independent variables are manipulated, their effect on one or more dependent variables is determined, and the levels of these independent variables are assigned according to the type of tests being performed [37]. Thus, manipulation of data is essential in a true experiment from which one may be able to infer cause and effect.

The aim of this experimental design is to shed some light on how various heuristics perform with regard to specific jobshop situations. In order to evaluate their impact, dependent variables, performance measures, and independent factors need to be identified. The choice of performance measure is left to the user to chose the one in which applies to his / her needs. A list of
performance measures can be found in French [6]. Independent factors, on the other hand are selected based on their effects on the performance measure.

The design stage is a significant part of the experiment. Considerations of how large a difference is to be detected, how much variation is present, and what size risk are to be tolerated are all important in deciding on the sample size. Since larger sample size means longer CPU time, the sample size is left to the user to chose depending on the computer resources available. Of course, the larger the sample size, the more accurate and reliable the results will be. Also important is the order in which the experiments are to be run, whether they be set at fixed levels or chosen randomly [38]. Once a decision has been made to control certain variables at specified levels, there are always a number of other variables that cannot be controlled. Randomization of the order will tend to average out the effect of these uncontrolled variables. However, for some statistical tests, it is required to have a balanced data. In such circumstances the order of runs is forced to be fixed.

Once the factors, performance measure, and number of runs have been decided on, the experimental data can be gathered. Depending on the number of runs, the jobshop is scheduled according to the heuristics picked for evaluation. At each run the values of each factor and the resultant
performance measure is saved in an output file. Thus, it is this file which contains the experimental data.

5. ANALYSIS OF DATA:

A mathematical model should describe the experiment. This model will show the response variable as a function of all factors to be studied and any restriction imposed on the experiment as a result of the method. Since the objective of the research project is to shed light on the stated problem, the problem should be expressed in terms of hypothesis. A research hypothesis states what the experimenter expects to find in the data [38].

Analysis includes the procedure for data collection, data reduction, and the computation of certain test statistics to be used in making decisions about various aspects of an experiment. Analysis involves the computation of test statistics and their corresponding decision rules for testing a hypothesis about the mathematical model. Once the test statistics have been computed, decisions can be made. These decisions should be made in terms that are meaningful to the experimenter [37].

The principle methods of analysis for experiment data is analysis of variance (ANOVA). ANOVA is a mathematical procedure for testing hypothesis. For further insight, additional data analysis and tests can be performed on the experimental data. One such test is Duncan's Multiple
Range Test. This test provides more support to conclusions made based on the size and the significance of individual factors and their interactions.

Several statistical analysis computer packages are available on the IBM main frame such as SAS, BMDP, SPSX, and others. For the purposes of our data analysis, Statistical Analysis System (SAS) was chosen for its availability and ease of use. The design of the experiment incorporated in the evaluation aid calls for analysis of variance and Duncan's multiple range test to be performed on the experimental data. As part of the package, a program representing the experimental model is written in SAS. This program will conduct the analysis of variance and Duncan's multiple range test. It can be modified to other models fairly easy.

The data gathered from this program is in tabular form. Several programs are written in software package TELAGRF to draw the one and two way interaction of factors. Decisions can be made based on strong statistical basis about the heuristics performance from these graphs.

Therefore, the method developed here for evaluation of jobshop scheduling heuristics can be applied to any jobshop situation in the industry. The only requirement is to provide the jobshop specification in order to model that. The validity of the method is illustrated through an example in the following chapter. For step by step
guide lines of the software package refer to Appendix A.
CHAPTER III

EXAMPLE PROBLEM

In the previous two chapters the need for a scheduling heuristics evaluation method and an approach to meet this need was discussed. This chapter will provide a sample problem in order to verify and illustrate the software package methodology developed.

The sample problem was generated by modeling a jobshop, scheduling the jobshop, and analyzing heuristics performance. The shop parameters are provided by the user. These parameters include machine processing time distribution (mean and variance) and number of machines. The parameters for this example is given in table 1. The resultant jobshop is presented in table 2. To schedule this jobshop three heuristics (RANDOM, SPT, FCFS) were used. They were included as part of the package for illustrative purposes.

An experimental design was conducted to evaluate heuristics performance. The criteria for this experiment was performance measure, maximum completion time (Cmax). Since the objective is to study the effect of these heuristics on the performance measure, heuristics were considered as one factor. Two other factors were selected based on the assumption that they affect completion time.
TABLE 1. SHOP PARAMETERS

***** GIVEN INPUT *****

NUMBER OF JOBS IS : 5
NUMBER OF MACHINES IS : 5

***** NUMBER OF OPERATION FOR EACH JOB *****

<table>
<thead>
<tr>
<th>JOB NO.</th>
<th>NO. OF OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

*** PROCES. TIME DIST. OF EACH MACH. FOR SMALL LOTS ***

<table>
<thead>
<tr>
<th>MACHINE NO.</th>
<th>MEAN</th>
<th>ST_DEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.00</td>
<td>3.00</td>
</tr>
<tr>
<td>2</td>
<td>25.00</td>
<td>6.00</td>
</tr>
<tr>
<td>3</td>
<td>17.00</td>
<td>2.00</td>
</tr>
<tr>
<td>4</td>
<td>33.00</td>
<td>8.00</td>
</tr>
<tr>
<td>5</td>
<td>65.00</td>
<td>11.00</td>
</tr>
</tbody>
</table>

** PROCES. TIME DIST. OF EACH MACH. FOR LARGE LOTS **

<table>
<thead>
<tr>
<th>MACHINE NO.</th>
<th>MEAN</th>
<th>ST_DEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.00</td>
<td>4.00</td>
</tr>
<tr>
<td>2</td>
<td>30.00</td>
<td>6.00</td>
</tr>
<tr>
<td>3</td>
<td>19.00</td>
<td>3.00</td>
</tr>
<tr>
<td>4</td>
<td>40.00</td>
<td>9.00</td>
</tr>
<tr>
<td>5</td>
<td>80.00</td>
<td>12.00</td>
</tr>
<tr>
<td>FOR JOB 1:</td>
<td>OPERATION</td>
<td>MACHINE</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>26.62</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>14.70</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>27.62</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>48.75</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>82.72</td>
</tr>
<tr>
<td>FOR JOB 2:</td>
<td>OPERATION</td>
<td>MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>66.29</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>41.74</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>36.21</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>20.20</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>20.27</td>
</tr>
<tr>
<td>FOR JOB 3:</td>
<td>OPERATION</td>
<td>MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>21.48</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>28.24</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>71.56</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>12.04</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>40.44</td>
</tr>
<tr>
<td>FOR JOB 4:</td>
<td>OPERATION</td>
<td>MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>81.66</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>43.85</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>12.79</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>18.66</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>30.10</td>
</tr>
<tr>
<td>FOR JOB 5:</td>
<td>OPERATION</td>
<td>MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>32.66</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>39.24</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>12.07</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>74.98</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>17.08</td>
</tr>
</tbody>
</table>
They are job / machine ratio and lot size. Table 3 presents a schedule and the resultant performance measure. A detailed discussion of all factors and performance measure will follow.

A. PERFORMANCE MEASURE : Cmax

The performance measure selected to illustrate the method is maximum completion time. Maximum completion time is the time at which processing of the last job on the last machine is completed. Minimizing the Cmax says that the cost of a schedule depends on how long the processing system is devoted to the entire set of jobs. Therefore, the most desirable schedule is the one with smallest Cmax, and thus the heuristic applied on this schedule is considered the best.

B. FACTOR 1 : HEURISTIC

The scheduling of operations at each machine is the most important factor in producing effective and near perfect schedules. For this reason, many different heuristics have been developed to be used at this stage of the scheduling. Since this study is to develop a methodology to evaluate these heuristics, we will conduct the experiment using three of these priority rules and later others can be added or changed. A description of the three heuristics used are as follows.

1.) RANDOM : Select an operation at random to be
TABLE 3. JOBSHOP SCHEDULE

FOR MACHINE 1:

<table>
<thead>
<tr>
<th>JOB</th>
<th>OPERATION</th>
<th>START TIME</th>
<th>FINISH TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>48.10</td>
<td>62.80</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>137.84</td>
<td>149.88</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>162.33</td>
<td>174.40</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>201.04</td>
<td>221.24</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>374.88</td>
<td>387.67</td>
</tr>
</tbody>
</table>

FOR MACHINE 2:

<table>
<thead>
<tr>
<th>JOB</th>
<th>OPERATION</th>
<th>START TIME</th>
<th>FINISH TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>21.48</td>
<td>49.73</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>62.80</td>
<td>90.42</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>90.42</td>
<td>123.09</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>123.09</td>
<td>164.83</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>406.35</td>
<td>436.45</td>
</tr>
</tbody>
</table>

FOR MACHINE 3:

<table>
<thead>
<tr>
<th>JOB</th>
<th>OPERATION</th>
<th>START TIME</th>
<th>FINISH TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>0.00</td>
<td>21.48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>21.48</td>
<td>48.10</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>249.37</td>
<td>266.45</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>266.45</td>
<td>286.73</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>387.67</td>
<td>406.35</td>
</tr>
</tbody>
</table>

FOR MACHINE 4:

<table>
<thead>
<tr>
<th>JOB</th>
<th>OPERATION</th>
<th>START TIME</th>
<th>FINISH TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>123.09</td>
<td>162.33</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>164.83</td>
<td>201.04</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>201.04</td>
<td>241.47</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>241.47</td>
<td>290.22</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>331.03</td>
<td>374.88</td>
</tr>
<tr>
<td>JOB</td>
<td>OPERATION</td>
<td>START TIME</td>
<td>FINISH TIME</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.00</td>
<td>66.29</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>66.29</td>
<td>137.84</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>174.40</td>
<td>249.37</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>249.37</td>
<td>331.03</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>331.03</td>
<td>413.75</td>
</tr>
</tbody>
</table>

C_MAX = 436.45
assign to the machine.

2.) SPT : Select the operation with the shortest processing time.

3.) FCFS : Select the operation that comes first to the current conflict machine, i.e., the one that finishes its preceding operation first at some other machine.

Heuristic is a qualitative factor and its levels are fixed.

C. FACTOR 2 : JOB / MACHINE RATIO

In evaluating various heuristics, we like to know whether they perform the same regardless of the job/machine ratio and how this effects the completion time. For this reason, the job/machine ratio is considered an important factor in deciding the best heuristic to use in scheduling a jobshop. For generating the simulated jobshop, one requirement was for the user to provide the number of machines and their processing time distribution. Therefore, the number of machines is fixed by the user. The number of jobs is a ratio [1-10] of fixed number of machines. The job/machine ratio is a quantitative factor and its levels are chosen randomly.
D. FACTOR 3 : LOT SIZE

One of the difficulties in comparing the results for different schedules has been the variety of job sizes to which scheduling heuristics have been applied. To discriminate the affect of the lot size from the effect of heuristics on the completion time, lot size is selected as separate factor. This study tests scheduling heuristics on two distinct set of job sizes, representing "small" lot size with jobs of small processing time and "large" lot size with jobs of large processing time.

For each machine, two sets of processing time distributions are assigned. The processing time for each job set in the experiment is randomly generated according to the lot size assigned.

The three factors chosen for the experiment provide for a complete factorial design in that all possible combinations are considered. A mathematical model is specified for the experimental design of this study. The model is

\[
Y_{ijk} = U + A_i + B_j + C_k + AB_{ij} + AC_{ik} + BC_{jk} + ABC_{ijk} + \varepsilon_{ijk}
\]

\(i = 1, 2, 3, \ldots, 10\) for the ten levels of ratio
\(j = 1, 2\) for the two lot sizes
\(k = 1, 2, 3\) for the three levels of heuristics
The dependent variable is Cmax and the independent variables are lot size, job/machine ratio, and heuristic. To insure randomness and unbiased results, 5 basic observations are made of the dependent variable, Cmax, for each level of independent variable [24]. Since three factors are chosen and they have 2, 3, and 10 levels respectively, the total number of observations is 300. After each run, the values of all three factors and the performance measure are saved on a file. When all 300 runs are completed, the data is ready for analysis.

Heuristics are expected to have direct and indirect effects in the regression model where the dependent variable is a measure of completion time. The interaction between heuristics and lot size is of interest, since its effect is not clearly noticeable.

Analysis of variance (ANOVA) is a mathematical procedure for testing hypothesis such as:

$$H_0 : \mu_{r1} = \mu_{spt} = \mu_{flex}$$

$$H_1 : \mu_{r1} \neq \mu_{spt} \neq \mu_{flex}$$

The null hypothesis states that all three levels of heuristics have the same mean. The alternative hypothesis states that at least one of the three levels has a different mean. However, this is not enough information to be able to make decisions on the type of heuristics to use. Additional support is needed based on the size and
significance of the individual factors and their interaction. For this reason, Duncan's multiple range test is used.

The experimental data is processed by the SAS program to compute the ANOVA and Duncan's multiple range test. The results for all cases investigated are presented in tables and graphical form in the following pages. The tables are generated by SAS and the graphs are done by TELAGRAF.

Table 4 presents the results for analysis of variance. The ANOVA model and the main effects of ratio, lot size, and heuristics are significant at the 0.0001 level for the performance measure. The main effect ratio is shown to be highly significant. This is to be expected, for as the ratio increases, the completion time increases, having a linear effect on the Cmax. In addition, the interactions are also significant, not to the same degree, at 0.0001 level.

To shed more light, a detailed statistical analysis is presented for the completion time. A Duncan's multiple range test was performed at the 5% level for all main effects means and the interactions effect on Cmax. The Duncan's results are presented in tabular form and each interaction is presented graphically and discussed.

Tables 5, 6, and 7 show the Duncan's multiple range test for ratio, lot size, and heuristic respectively. As expected, when the ratio increased, so did the mean. From table 6, a similar relationship is observed; e.g., increased
TABLE 4. ANOVA

CLASS LEVEL INFORMATION

<table>
<thead>
<tr>
<th>CLASS LEVEL INFORMATION</th>
<th>CLASS</th>
<th>LEVELS</th>
<th>VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>RATIO</td>
<td>10</td>
<td>1 2 3 4 5 6 7 8 9 10</td>
<td></td>
</tr>
<tr>
<td>LOT</td>
<td>2</td>
<td>L S</td>
<td></td>
</tr>
<tr>
<td>HURESTIC</td>
<td>3</td>
<td>FCFS RND SPT</td>
<td></td>
</tr>
</tbody>
</table>

NUMBER OF OBSERVATIONS IN DATA SET = 300

DEPENDENT VARIABLE: Cmax

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>MEAN SQUARE</th>
<th>F VALUE</th>
<th>PR&gt;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL</td>
<td>59</td>
<td>458650740.2391</td>
<td>7773741.3599</td>
<td>350.51</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>240</td>
<td>1381477.684195</td>
<td>5756.1570174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>299</td>
<td>460032217.9233</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PR>F R-SQUARE MSE Cmax MEAN

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>ANOVA SS</th>
<th>F VALUE</th>
<th>PR&gt;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>RATIO</td>
<td>9</td>
<td>436004108.19538</td>
<td>8416.19</td>
<td>0.0001</td>
</tr>
<tr>
<td>LOT</td>
<td>1</td>
<td>15529919.099187</td>
<td>2697.97</td>
<td>0.0001</td>
</tr>
<tr>
<td>RATIO*LOT</td>
<td>9</td>
<td>3999307.2441247</td>
<td>77.20</td>
<td>0.0001</td>
</tr>
<tr>
<td>HURISTIC</td>
<td>2</td>
<td>2470267.3256464</td>
<td>214.56</td>
<td>0.0001</td>
</tr>
<tr>
<td>RATIO*HUR</td>
<td>18</td>
<td>537914.40228885</td>
<td>5.19</td>
<td>0.0001</td>
</tr>
<tr>
<td>LOT*HUR</td>
<td>2</td>
<td>18031.06350374</td>
<td>1.57</td>
<td>0.2110</td>
</tr>
<tr>
<td>RA<em>LOT</em>HU</td>
<td>18</td>
<td>91192.80897373</td>
<td>0.88</td>
<td>0.6032</td>
</tr>
</tbody>
</table>

==================================================================
TABLE 5. DUNCAN’S MULTIPLE RANGE TEST FOR RATIO

\[ \alpha = 0.05 \quad \text{df}=240 \quad \text{MSE}=5756.16 \]

Means with the same letter are not significantly different.

<table>
<thead>
<tr>
<th>DUNCAN GROUPING</th>
<th>MEAN</th>
<th>N</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4286.5</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>3659.5</td>
<td>30</td>
<td>9</td>
</tr>
<tr>
<td>C</td>
<td>3424.3</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>2967.4</td>
<td>30</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>2581.9</td>
<td>30</td>
<td>6</td>
</tr>
<tr>
<td>F</td>
<td>2177.9</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>1768.7</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>H</td>
<td>1335.3</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>I</td>
<td>925.3</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>J</td>
<td>485.1</td>
<td>30</td>
<td>1</td>
</tr>
</tbody>
</table>
TABLE 7. DUNCAN'S MULTIPLE RANGE TEST FOR HEURISTIC

\[ \alpha = 0.05 \quad DF=240 \quad MSE=5756.16 \]

Means with the same letter are not significantly different.

<table>
<thead>
<tr>
<th>DUNCAN GROUPING</th>
<th>MEAN</th>
<th>N</th>
<th>HEURISTIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2496.9</td>
<td>100</td>
<td>SPT</td>
</tr>
<tr>
<td>B</td>
<td>2362.3</td>
<td>100</td>
<td>RND</td>
</tr>
<tr>
<td>C</td>
<td>2274.6</td>
<td>100</td>
<td>FCFS</td>
</tr>
</tbody>
</table>

TABLE 6. DUNCAN'S MULTIPLE RANGE TEST FOR LOT SIZE

\[ \alpha = 0.05 \quad DF=240 \quad MSE=5756.16 \]

Means with the same letter are not significantly different.

<table>
<thead>
<tr>
<th>DUNCAN GROUPING</th>
<th>MEAN</th>
<th>N</th>
<th>LOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2612.1</td>
<td>150</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>2157.1</td>
<td>150</td>
<td>S</td>
</tr>
</tbody>
</table>
lot size causes an increase in mean. One not so obvious fact is seen in table 7 where group C, first come first serve, seems to offer the best mean. And SPT (shortest processing time) assignment delivers the worse. Looking at this table alone, it can be concluded that FCFS should be employed for both large and small lot size sets of jobs and given job / machine ratios within the range.

**E. LOT-RATIO INTERACTION :**

Table 8 and figure 1 illustrate the interaction of factor lot size with factor ratio. For both levels of lot size, Cmax increased as ratio climbed. But the difference between the two levels is much more noticeable at higher ratios. This is a normal result considering the linear relationship between the two.

**F. LOT-HEURISTIC INTERACTION :**

A plot of the first order interaction between lot size and heuristic is shown in figure 2. Data used to plot this figure is from table 9. For both small lot size and large lot size FCFS performs better than the other two. However, RND is easier to apply and take less CPU time; therefore, one might sacrifice a bit in efficiency but reduce the cost of scheduling.

On the other hand, the plot shows that the difference in performance of SPT is very significant regardless of
### TABLE 8. RATIO LOT INTERACTION

<table>
<thead>
<tr>
<th>RATIO</th>
<th>LOT</th>
<th>N</th>
<th>CMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L</td>
<td>15</td>
<td>542.79066</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>15</td>
<td>435.45200</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>15</td>
<td>1011.09123</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>15</td>
<td>839.47609</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>15</td>
<td>1475.08133</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>15</td>
<td>1195.51200</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>15</td>
<td>1537.39533</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>15</td>
<td>1600.01133</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>15</td>
<td>2360.52467</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>15</td>
<td>1995.42533</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>15</td>
<td>2620.23333</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>15</td>
<td>2343.57133</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>15</td>
<td>3293.21267</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>15</td>
<td>2701.60333</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>15</td>
<td>3747.57267</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>15</td>
<td>3101.01333</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>15</td>
<td>4235.44267</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>15</td>
<td>3463.47533</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>15</td>
<td>4697.98867</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>15</td>
<td>3875.08867</td>
</tr>
</tbody>
</table>

### TABLE 9. LOT HURISTIC INTERACTION

<table>
<thead>
<tr>
<th>LOT</th>
<th>HURISTIC</th>
<th>N</th>
<th>CMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>FCFS</td>
<td>50</td>
<td>2492.99320</td>
</tr>
<tr>
<td>L</td>
<td>RND</td>
<td>50</td>
<td>2619.57920</td>
</tr>
<tr>
<td>L</td>
<td>SPT</td>
<td>50</td>
<td>2723.74940</td>
</tr>
<tr>
<td>S</td>
<td>FCFS</td>
<td>50</td>
<td>2056.26680</td>
</tr>
<tr>
<td>S</td>
<td>RND</td>
<td>50</td>
<td>2144.93740</td>
</tr>
<tr>
<td>S</td>
<td>SPT</td>
<td>50</td>
<td>2269.98380</td>
</tr>
</tbody>
</table>
FIGURE 2. LOT-HEURISTIC INTERACTION

Legend:
- LARGE LOT
- SMALL LOT

HEURISTIC (1 = FCFS 2 = RND 3 = SPT)
lot size. Again, by looking at this figure, it seems FCFS is the most promising of the three.

6. RATIO-HEURISTIC INTERACTION;

The first order interaction between ratio and heuristic is illustrated in figure 3 and table 10. There is no significant difference in the mean of $C_{max}$ between the RND, SPT, and FCFS as the ratio increases. However FCFS produces the best $C_{max}$ of all three.

H. LOT-RATIO-HEURISTIC INTERACTION;

The second order interaction between lot size, ratio, and heuristic proves to be significant to the Duncan's multiple range test and is shown in table 11 and figures (4-10). Figures 6, 7, 8 plot the lot-ratio interaction for each level of heuristic, figures 4 and 5 plot heuristic ratio interaction for each level of lot size, and figures 9, and 10 plot the heuristic lot interaction for two levels of ratio.

The analysis of second order interaction effects provides the following additional information which is not evident from first order interaction. When same number of machines and jobs are present, RND clearly gives the worst scheduling heuristic.

In conclusion, from the data gathered and analysis conducted for this simulated jobshop it seems that FCFS is
<table>
<thead>
<tr>
<th>RATIO</th>
<th>HURISTIC</th>
<th>N</th>
<th>CMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FCFS</td>
<td>10</td>
<td>436.23500</td>
</tr>
<tr>
<td>2</td>
<td>RND</td>
<td>10</td>
<td>527.03000</td>
</tr>
<tr>
<td>3</td>
<td>SPT</td>
<td>10</td>
<td>504.08000</td>
</tr>
<tr>
<td>2</td>
<td>FCFS</td>
<td>10</td>
<td>825.98800</td>
</tr>
<tr>
<td>3</td>
<td>RND</td>
<td>10</td>
<td>928.70300</td>
</tr>
<tr>
<td>2</td>
<td>SPT</td>
<td>10</td>
<td>1021.16000</td>
</tr>
<tr>
<td>3</td>
<td>FCFS</td>
<td>10</td>
<td>1259.60800</td>
</tr>
<tr>
<td>3</td>
<td>RND</td>
<td>10</td>
<td>1363.87700</td>
</tr>
<tr>
<td>3</td>
<td>SPT</td>
<td>10</td>
<td>1382.40500</td>
</tr>
<tr>
<td>4</td>
<td>FCFS</td>
<td>10</td>
<td>1668.43800</td>
</tr>
<tr>
<td>4</td>
<td>RND</td>
<td>10</td>
<td>1802.21300</td>
</tr>
<tr>
<td>4</td>
<td>SPT</td>
<td>10</td>
<td>1835.45900</td>
</tr>
<tr>
<td>5</td>
<td>FCFS</td>
<td>10</td>
<td>2072.94800</td>
</tr>
<tr>
<td>5</td>
<td>RND</td>
<td>10</td>
<td>2185.15500</td>
</tr>
<tr>
<td>5</td>
<td>SPT</td>
<td>10</td>
<td>2275.52200</td>
</tr>
<tr>
<td>6</td>
<td>FCFS</td>
<td>10</td>
<td>2482.65500</td>
</tr>
<tr>
<td>6</td>
<td>RND</td>
<td>10</td>
<td>2562.08000</td>
</tr>
<tr>
<td>6</td>
<td>SPT</td>
<td>10</td>
<td>2700.97200</td>
</tr>
<tr>
<td>7</td>
<td>FCFS</td>
<td>10</td>
<td>2879.06100</td>
</tr>
<tr>
<td>7</td>
<td>RND</td>
<td>10</td>
<td>2993.33100</td>
</tr>
<tr>
<td>7</td>
<td>SPT</td>
<td>10</td>
<td>3119.83200</td>
</tr>
<tr>
<td>8</td>
<td>FCFS</td>
<td>10</td>
<td>3309.40100</td>
</tr>
<tr>
<td>8</td>
<td>RND</td>
<td>10</td>
<td>3373.11500</td>
</tr>
<tr>
<td>8</td>
<td>SPT</td>
<td>10</td>
<td>3590.36300</td>
</tr>
<tr>
<td>9</td>
<td>FCFS</td>
<td>10</td>
<td>3696.92500</td>
</tr>
<tr>
<td>9</td>
<td>RND</td>
<td>10</td>
<td>3841.11900</td>
</tr>
<tr>
<td>9</td>
<td>SPT</td>
<td>10</td>
<td>4040.33000</td>
</tr>
<tr>
<td>10</td>
<td>FCFS</td>
<td>10</td>
<td>4115.03800</td>
</tr>
<tr>
<td>10</td>
<td>RND</td>
<td>10</td>
<td>4245.96000</td>
</tr>
<tr>
<td>10</td>
<td>SPT</td>
<td>10</td>
<td>4498.54300</td>
</tr>
</tbody>
</table>
# Table 11: Ratio Lot Huristic Interaction

<table>
<thead>
<tr>
<th>RATIO</th>
<th>LOT</th>
<th>HURISTIC</th>
<th>N</th>
<th>CMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>465.88600</td>
</tr>
<tr>
<td>1</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>611.02200</td>
</tr>
<tr>
<td>1</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>551.43200</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>FCFS</td>
<td>5</td>
<td>406.59000</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>RND</td>
<td>5</td>
<td>443.03800</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>SPT</td>
<td>5</td>
<td>456.72800</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>902.66400</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>1060.21400</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>1070.39600</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>FCFS</td>
<td>5</td>
<td>749.31200</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>RND</td>
<td>5</td>
<td>797.19200</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>SPT</td>
<td>5</td>
<td>971.92400</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>1390.24000</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>1515.82000</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>1519.18400</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>FCFS</td>
<td>5</td>
<td>1128.97600</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>RND</td>
<td>5</td>
<td>1211.93400</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>SPT</td>
<td>5</td>
<td>1245.62600</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>1835.25800</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>1990.61200</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>1986.31600</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>FCFS</td>
<td>5</td>
<td>1501.61800</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>RND</td>
<td>5</td>
<td>1613.81400</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>SPT</td>
<td>5</td>
<td>1684.60200</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>2265.82200</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>2378.92000</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>2436.23200</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>FCFS</td>
<td>5</td>
<td>1880.07400</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>RND</td>
<td>5</td>
<td>1991.39000</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>SPT</td>
<td>5</td>
<td>2114.81200</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>2729.48000</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>2773.46000</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>2957.76000</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>FCFS</td>
<td>5</td>
<td>2235.83000</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>RND</td>
<td>5</td>
<td>2350.70000</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>SPT</td>
<td>5</td>
<td>2444.18400</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>FCFS</td>
<td>5</td>
<td>3171.43600</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>RND</td>
<td>5</td>
<td>3287.66400</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>SPT</td>
<td>5</td>
<td>3420.53800</td>
</tr>
</tbody>
</table>
### Table 11. (Cont.) Ratio Lot Huristic Interaction

<table>
<thead>
<tr>
<th>Week</th>
<th>Method</th>
<th>Mean Completion Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>S</td>
<td>FCFS</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>RND</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>SPT</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>FCFS</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>RND</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>SPT</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>FCFS</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>RND</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>SPT</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>FCFS</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>RND</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>SPT</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>FCFS</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>RND</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>SPT</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>FCFS</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>RND</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>SPT</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>FCFS</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>RND</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>SPT</td>
</tr>
</tbody>
</table>
FIGURE 4. RATIO-LOT-HEURISTIC INTERACTION AT SMALL LOT SIZE
Figure 5. Ratio-Lot-Heuristic interaction at (large lot) of lot size

Legend: X = Ratio 1, ○ = Ratio 0.5, + = Ratio 0.25, △ = Ratio 0.125, × = Ratio 0.0625

Heuristics: 1 = FCFS, 2 = RND, 3 = SPT

C-Max Mean

5000 4000 3000 2000 1000 0
FIGURE 7. RATIO-LOT-HEURISTIC INTERACTION AT (RND) LEVEL OF HEURISTIC

Legend
X LARGE LOT
O SMALL LOT
FIGURE 10. LOT-RATIO-HEURISTIC INTERACTION AT SECOND LEVEL OF RATIO

Legend
× LARGE LOT
○ SMALL LOT

HEURISTICS (1=FCFS, 2=RND, 3=SPT)
the heuristic with potential and should be given more consideration; but SPT should most definitely be avoided.
CHAPTER IV

CONCLUSION

Analytic approaches seeking optimal solution to jobshop scheduling problems have, so far, proved to be computationally infeasible. The heuristics approach using priority rules, on the other hand, sacrifice optimally but are feasible and have been widely employed. The purpose of this research was to develop a new approach to evaluate scheduling heuristics. The final product of this system design is in part simulation, application, and experimental analysis.

The package has met all of the initial requirements established for its design. In addition, it has provided a number of features that were not in the initial design.

A simulation model of a generated jobshop was used to produce the example problem. A multifactor ANOVA model was design to evaluate the impact of factors on the performance measure.

The appropriate statistical test conducted on the simulation results supports the fact that use of different heuristics does result in a significantly different performance measure and mean completion time. Of the heuristics used in the simulation process, FCFS (first come first serve) was found to exhibit the best performance in all situations but one.
These results and conclusions are about a set of specific jobshops. However, what makes this study differ from the rest is its flexibility. The same simulation techniques and analysis can be performed on any existing jobshop problem in the industries.

Further research should be done on this problem in connection with the sequence programming method, and in the context of the general jobshop scheduling. At present, heuristic approach seems to hold the most hope for scheduling problem. This fact should be looked at more closely and methods be developed employing various kind of heuristics. These methods ought to be flexible so that industries of various nature could apply their jobshops and decrease scheduling cost.
BIBLIOGRAPHY


2.) J. G. Kimemia, " Hierarchical control of Production in Flexible Manufacturing system, "


23.) J. S. Fryer, " Operating Policies in Multiechelon Dual Constrained Job Shops," *Management Science*


36.) M. R. Carey and D. S. Johnson, "Strong


INTRODUCTION:

The package is part simulated and part application and analysis that might be used for jobshop generation, scheduling an actual jobshop, or investigation on which type of dispatching rules to used. It is written for IBM mainframe and was developed at Ohio University in College of Engineering and Technology’s Department of Industrial and System Engineering.

The package is especially designed for user who have no knowledge of computers or computer programming. For this reason, the user just needs to supply the required informations.

Users of the package are assumed to have a working knowledge of jobshop and scheduling. Scheduling theory is not explained within the package, as it is to be used as a tool, not a teacher. In addition, until the user becomes familiar with the package, it is suggested that the information contain in this report be read carefully.

A.) LOGON:

To log on the IBM mainframe, you must first have an account number for the system. This account may be obtained from the computer center at Haning Hall.
To log on, make sure your terminal is on. Next press the break key several times, and then RETURN key until you get the system prompt:

ENTER CLASS

Type one and then press the RETURN key. Now you are asked to log on. Just type:

LOGON DEPT. MACHINE NUMBER

Each department has its own virtual machines. You need to specify the department and machine number. Now you are asked to enter the password. At this point, type in your assigned password. If you entered your password correctly, you will be asked to supply your account number. After this, you type your account number and press the RETURN key. Then a message will appear on the screen asking you to input a name for your output listing. Enter a six letter name and press the RETURN key. The system prompt, which is (CMS), appear. You are ready to run the package.

B.) RUNING THE PACKAGE:

Depending on whether you wish to run the package on the interactive or batch mode, the following needs to be done. To run the package as the latter, the necessary job cards must be specified at the begging of the program, and the input data must be specified at the end of the program. To run the package interactive, just enter
To run an interactive, you just load the package to your disk and start execution. The command to do this is:

RPASCAL HEA

This command compiles the program and starts execution. When system returns

EXECUTION BEGAN

Just hit return and you are asked to input data. Please follow the instructions on screen and provide the appropriate data points, then press RETURN key twice after every input. You are asked to give the number of replication. These parameters include the number of machines and processing time distribution (mean and variance). There are two set of processing time distribution for each machine, small lots and large lots. At this point you are asked if you wish to see the jobshop generated and the schedules created. If the answer is "yes", you will see the jobshop and schedules flash through the screen very fast. Thus, if you like to have the copy of the schedules and the jobshop, it is best to specify the file definition and get a hard copy.

FI SYSTEM PRINTER

This command will used your output to the printer. It should be specified before the program HEA is run.

A massage will appear on the screen after every
replication in the program is being run. It is recommended not to use interactive if number of replications is greater than one due to time constraint. Instead the program should be run on BATCHMON where the user is free to do other work. The command to enter for batchmon is

```
SUB HEAB PASCAL BATCHMON
```

when the execution is finished the results is send back to the reader of that particular virtual machine.

The FILE SCHEDULE contains the experimental data regardless of the type of run (interactive or batchmon). This data is supplied to the SAS program for analysis.

C.) RUN SAS PROGRAM:

After the SAS program is recalled to the A disk enter command

```
E ANALYZE SAS
```

In the line after `// * SCHEDULE type`

```
GET FILE SCHEDULE
```

then type `FILE` to exit the EDIT mode. Then, in CMS mode, run the SAS program by entering

```
SUB ANALYZE SAS
```

The results from SAS program is in tabular form. The ANOVA and Duncan's mean test can be analyzed from these tables. However, interactions are not so obvious. Therefore the data for the interactions is run and analyzed
by the TELAGRAF program to generate graphs.

D.) RUN TELAGRAF PROGRAM:

There are ten programs which graphs the one and two way interaction. The following instructions can be used to run any one of them.

In CMS mode type TELAGRAF. The system will return a massage, SPECIFY FILE. You enter the name of the program you wish to run. Again the system will return with a massage GENERATE LEVEL ....ENTER. You need to type

    GEN A PLOT

System will return massage ENTER. You type GO. The graph is generated on the screen after a few seconds. To get a print out of the graph:

1) press setup key
2) set F6 to graph
3) press setup key
4) press PRINT key
5) press setup key
6) set F6 to auto
7) press F7
8) type QUIT.

At this point you are back in the CMS mode and can start from top to the rest of the programs for information. After all the graphs are made, decisions can be made base on these graphs about which heuristics to use.
APPENDIX B

Documentation
DOCUMENTATION

To give sufficient documentation for the package, the following two distinct areas are provided.

1.) USER DOCUMENTATION: To assist the user in running the package, a user manual was provided. This manual explains the procedure for loading the computer and information necessary to start execution of the package.

2.) PROGRAM DOCUMENTATION: For further modification and addition to the package, it was necessary to provide documentation on the design and coding of the program in the package.

In order to provide an explanation about the program, extensive use of comments was made in the program. Descriptive explanations were provided for strategic sections of each procedure. A general explanation of the program’s function is also given at the beginning of each procedure. Also, the listing of the program is supplied in appendix E.

A programmer’s reference manual was provided in appendix D for future modification and with information on the design of the program. This manual explains the function of the various variables used in the procedures of the package.

Finally, to provide a visual representation of the logic flow in each procedure, flowcharts were produced.
These flowcharts provide a brief description of the structure, flow, and the individual procedure. A detailed description of function of each procedure follows:

1.) DATA INPUT:
   
i.) REQUIREMENTS: The user of package must be able to create files into which data will be stored for use by the other procedures of the package. The user should be able to enter number of machines and their processing time distribution for both large and small lot size. The job machine ratio needs to be generated and the number of jobs determined. Also, for each job the number of operations needs to be assigned and each has to be processed on a machine only once. There must be at least as many machines as maximum number of operations.

   ii.) DESIGN: The procedure that provides the input of data, IN-DATA, allows the user to specify the input device to be used. When input is from external files, the file definitions must be specified by the user and a file needs to be created and each data point be in one line. The first data point is the number of repetition to be performed. The second data point is the number of machines which are going to be used on this jobshop. Next section of input is the processing time distribution of the machines. The user is asked to enter two distributions for each machine, one for large lots and one for small lots. Using
the number of machines given and the two functions, random number generator (RANDOM) and random integer generator (RNDINT), the job machine ratio is generated.

2.) ASSIGN OPERATION TO MACHINES:

i.) REQUIREMENT: The operation being assigned to any machine must be assumed initially an equal likelihood. But the program must be flexible to be modified to any distribution required by the user. In addition, no one machine can be assign to more than one operation of any job.

ii.) DESIGN: ASSIGN-OPER-TO-MACH is the procedure that provides assignment capability in the package. This procedure uses the following two functions to meet the above requirements, the random number generator function (RANDOM) and the random integer generator function (RNDINT). The process of assignment is as follows. First, the number of operations for every job is noted. Second, for any one of these operations a machine is randomly chosen. This machine should not been assigned to any other operations of this job. This process continues until all operations of all jobs are assigned to a machine. To insure against biased assignment, two ways of randomization are applied. Therefore, both machines and operations are randomly chosen.
3.) ASSIGN PROCESSING TIME TO OPERATION:

I.) REQUIREMENTS: Each machine has a processing distribution time. The distribution must be specified by the user. Mean and standard division of an assumed normal distribution are statistic parameters which are needed. Also, for each machine the user needs to provide two different processing times for "large" and "small" lot size.

II.) DESIGN: PROCESS-TIME is the procedure that allows the assignment of processing time to operations. The technique used to achieve this objective is a scheme employing random numbers, that is, $u(0,1)$ random variables which are used for solving deterministic problems. Under this technique, various functional relationships are used in order that certain complicated probability design functions can be represented in a similar form which deviates can easily be generated by the more common technique [33]. The normal distribution is one of the most widely used in applied statistics and has two parameters; $\mu$ = expected value and $\sigma^2$ = variance. The cumulative distribution function of a normal variable $X$ is given by

$$p(x) = \int_{-\infty}^{\infty} \frac{1}{\sigma \sqrt{2\pi}} e^{-\left[\frac{1}{2}(x-\mu)^2/\sigma^2\right]} \, dx$$

where $\infty > x > -\infty$. 
This function cannot be evaluated analytically. Hence, many of the methods for generating normal deviates use approximation methods. If the parameters $\mu$ and $\sigma^2$ have values of zero and 1, respectively, the distribution is known as the standard normal density with

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \quad -\infty < z < \infty$$

Any normal variate $X$ with parameters $\mu$ and $\sigma^2$ can be converted to a standard normal variate $Z$ by using

$$Z = (X - \mu) \sigma$$

Conversely, in order to convert a standard normal deviate $Z$ to a normal deviate with parameters $\mu$ and $\sigma^2$, must use

$$X = \sigma Z + \mu$$

Hence, to generate a normal deviate $X$ with parameters, we only need to generate a standard normal deviate and apply the preceding equation. The method used for generating standard normal deviates is approximation techniques [39]. The approximation methods are all related to the inverse transform technique and are used when exact methods are impossible. Here, the approximation method is used to approximate the inverse transform to the cumulative distribution function of the distribution to be simulated.

Now to generate a deviate from the standard normal
density function given by

\[ f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \quad -\infty \leq x \leq \infty \]

An approximation has been given by Kahn.

\[ -\frac{x^2}{2} e^{-\frac{x^2}{2}} \approx \frac{2 e^{-\frac{x^2}{2}}}{(1 + e^{-\frac{x^2}{2}})^2} \quad x > 0 \quad \text{and} \quad \kappa = \left( \frac{3}{\pi} \right)^{\frac{1}{2}} \]

The cumulative distribution function to the approximation is calculated as

\[ F(x) = \frac{2}{1 + e^{-\kappa x}} - 1 \]

The inverse to this approximation cumulative distribution function is then

\[ Z = X = \frac{1}{\kappa} \ln \left( \frac{1 + R}{1 - R} \right) \]

A random sign is then attached to this variate where \( R \) is a deviate from the uniform distribution. Kahn's approximation derives a normal random deviate directly from an approximation to the density function, \( F(x) \).
functions ST-NORMAL and RANDOM employed the above logic to determine the processing time of the operations. Depending on the lot size of each set of jobs, the appropriate machine processing time distribution is used to calculate the processing time of each operation.

4.) OPERATION PROCESSING ORDER FOR EACH JOB:

i.) REQUIREMENT: The randomly chosen operation must be on an equal likelihood basis. The final jobshop must have its operations in ascending order for each job.

ii.) DESIGN: In the simulation of operations, the random number generator, function RANDOM, and the random integer generator, function RNDINT, were used. Function RANDOM is based on uniform distribution. Therefore, the numbers generated by this function are chosen equally. The procedure SOR-OP is used to order the final jobshop according to its operations. The result after this procedure is a jobshop in which each job goes through a set of operations in an ascending order.

5.) PRINT OUT OF SIMULATED JOBSHOP:

i.) REQUIREMENTS: The output should specify the number of the jobs, the number of the machines, the processing time distribution of each machine, and the number of operations for each job. The simulated jobshop should be in order of jobs and for each job, the operation,
machines, and processing time.

ii.) DESIGN: To meet the requirements presented above, procedure PRINT-OUT was developed. This procedure allows the user to get a copy of the simulated jobshop and the given data.

6.) CONSTRUCTION OF SCHEDULABLE OPERATION LIST:

i.) REQUIREMENT: The schedule of operations in the algorithm should be one at a time. It should be determined which operation is schedulable. An operation is schedulable if all the operations which must precede it within its job have already been scheduled. Since there are NM ( N = number of jobs, M = number of machines ) operations, the algorithm should iterate through NM stages.

ii.) DESIGN: At the first stage, none of the operations of any jobs is scheduled. Therefore, there are no operations which must precede. In other words, at this stage the first operation of all jobs are schedulable. The information stored in the schedulable operations list includes the jobs and the operations of these jobs ready to be scheduled, the machines these operations are going to be processed by, and the start and finish time of these operations. The procedure that provides these functions is ADJUT. Moreover, this procedure sets the position of the scheduable operations and sets the stage.
7.) SCHEDULEING OF ONE OPERATION:

i.) REQUIREMENTS: Given the schedulable operation list, the operation with the earliest finish time should be found (the finish time of any operation is its start time plus processing time). If there are more than one schedulable operations with the earliest finish time, then all these operations must be maintained. In this case, one of the several operations must be chosen accordingly to the heuristic, where the heuristic needs to be randomly chosen for each schedule. This operation shall be scheduled and the finish time of the machine used for this operation must be remembered.

ii.) DESIGN: To meet the first requirement, the procedure SMALST-FINISH-TIME was developed. This procedure searches through the schedulable operation list and finds the operation with the earliest finish time. Then the index of this operation is kept for future use.

Procedure SCHUL-OPER fulfills rest of the requirements. By comparing the machine used on the operation with earliest finish time to the machines used on remaining schedulable operations, it determines if any one of them uses the same machine. In such case, all operations with same machine and their indices are stored for future use. If there are no other operations using same machine, the operation with earliest finish time is assigned to be
scheduled on this machine next. Otherwise, one must be chosen from the list of operations using the same machine as the operation with earliest finish time. The decision on which to choose depends on the heuristic. The type of heuristic is randomly picked by random number generator, RANDOM function. Three types of heuristics are available on this package, RANDOM (random selection of operation), SPT (shortest processing time), and FCFS (first come first served). When the SPT assignment is picked, procedure SPT is allocated. This procedure was developed to determine the operation with the shortest processing time. By comparing the processing time of all operations in the list of operations using same machine as the operation with the earliest finish time, this task is achieved. The index of this operation is returned to SCHUL-OPER to be used in the future.

Procedure FCFS was designed to provide FCFS assignments. To determine which operation has been in the list (the list of operations using the same machine as the operation with the earliest finish time) the longest time, the number of stages at which each operation has been schedulable is compared. The operation with the greatest number of stages is selected to be scheduled. The index of this operation is passed to procedure SCHUL-OPER.

In procedure SCHUL-OPER, the operation with that
index, along with the machine to be processed on, the job which it belong to and its start time and finish time, is filed into another list called partially scheduled. It is this file that at the end contains all the scheduled operations.

The last requirement of this section was to update the finish time of the machine used by the operation just scheduled. This is necessary so that next operation assigned to this machine would not overlap with the operation just scheduled. Therefore, the next time this machine is free will be set to the finish time of this operation.

8.) UPDATING SCHEDULABLE OPERATION LIST FOR NEXT STAGE.

i.) REQUIREMENTS: As one operation gets scheduled, the schedulable operation list needs to be updated. It should be checked whether any more operations are left from this job. If so, the next operation should replace the previous one in the schedulable operation list. Otherwise, this job should be eliminated from the list. In addition, the number of stages that each operation has been in the schedulable list must be incremented and the start and finish time of operations in this list must be updated according to time when their machines are free.

ii.) DESIGN: This section of the algorithm is most
crucial part. Therefore, a great deal of care and effort was put in its design. The accuracy of the results obtained from this package depends on how requirements are represented as a logical flow. Extensive testing was done to insure correct solution.

The procedure developed to provide these capabilities is PRPAR-NXT-STAG. To meet the first requirement, it compares the position of the operation just scheduled with number of operation that this job has. If this operation was the last operation of this job, then the two numbers should be eliminated from the schedulable operations list. Otherwise, the just scheduled operation should be replaced by the next operation of this job. Using the position on which the previous operation was fetched from the operation in the next position, the machine it is to be processed on, start and finish time, and the processing time are sent to the schedulable operation list. The number of stages for this operation is set to one and its position is incremented.

With the exception of the operation just scheduled and the jobs eliminated from the list, the remaining schedulable operations enter a new stage; as a result, their stage number is incremented. The last requirement was to update the start and finish time of each operation. To insure this, for every machine, the schedulable operation list is searched. The start time of these operations is compared to the time
when the machine used on these operations is next free. If the machine is free before operation starts, nothing is changed. Otherwise, the time this machine is free has to be the new start time for that particular operation. Thus, the finish time is also adjusted accordingly.

9.) FINAL SCHEDULE IN ORDER OF MACHINE:

   i.) REQUIREMENT: In order to graph a Gantt diagram of the final schedule, the partial schedule list must be sequenced according to machines.

   i.) DESIGN: A Gantt diagram is a simple diagram used to plot the final schedule. From the Gantt diagram the idle time of machines and jobs can be observed as well as the completion time. To be able to use a Gantt diagram, the schedule has to be in order of machines, so that the time a machine is used can be drawn. Procedure SORT provided this ability and stored the final schedule in order of machines. But to draw an accurate diagram, the operations assigned to this machine must be sorted in order of their start-time.

10.) COMPLETION TIME:

   i.) REQUIREMENT: For each schedule performance measure, completion time must be calculated.

   ii.) DESIGN: The performance measure, completion time,
is the time at which processing of the last job on the last machine is completed. This measure is computed by procedure FIND-C-MAX. By comparing the last operation of each job, the largest finish is found. This value is the completion time of this schedule.

ii.) OUTPUT FINAL SCHEDULE:

i.) REQUIREMENT: The output should specify machine number and for each machine, the jobs and their associated operation, operation start time, and finish time.

ii.) DESIGN: The final schedule should be readable and easy to follow by the user. From it a Gantt diagram will be drawn. To meet the requirements, procedure FINAL-SCHUL was developed. This procedure organizes the final schedule into tables, one for each machine.

12.) CREATE A FILE FOR SAS:

i.) REQUIREMENT: To analyze the schedules obtained from this algorithm, its criteria and performance measures need to be stored for every one of the schedules.

ii.) DESIGN: Procedure PRINT-SAS-DATA creates an external file to be used by SAS. For each run of the schedule, this procedure writes into the external file independent variables such as ratio, lot size and heuristic, plus the dependent variable C-MAX.
MAIN PROGRAM:

START

set last random no. to seed

input the no of operation

For the first opera

call in-data to input the initial data

call initialize to initialize the job shop matrix

A

A

call assign-oper-to-mach to generate the jobshop

call process-time to assign process time to each oper

call sort-op to sort initial problem in order of oper. for each job

call chose-hu to select the heuristic to be used

call schul to schedule the jobshop problem

END
MAIN0 PROGRAM:

START

- call procedure SAS to fill the initial jobshop matrix

- set stage to 1

- do
  - while no. stage ≤ total no. of jobs and mach
    - yes
      - call the procedure smallest-finish time to find the available operation with smallest finish time
    - no
      - call the procedure sort to order the schedule oper. in order of mach. no.
      - call the procedure find c-max to determine the completion time of this schedule
      - call schu-oper of another oper.
      - call PRP-NXT-STAG to update the list of available operation

output the final schedule

RETURN
PROCEDURE IN-DATA:

START

is the 1st run

yes

input no. of machine

no

input P.T. dist. for small lots

input P.T. dist. for large lots

determine the job-machine ratio

assign no. of job

assign no. of oper. for each job

STOP
PROCEDURE INITIALIZE :

START

1. take the first operation of this job

2. initialize the oper., mach., proc., time to zero

3. if this last oper. is no
   then return

4. is this last job
   yes
   then return
   no
   then
   if this last oper. is yes
   then return
   no
   then
   return
PROCEDURE SOR-OP:

START

take the first job and its operation

take the first operation

is first op. second op.

yes

SWAP the two oper.

no

next operation until finish

next job until finish

RETURN
PROCEDURE PRINT-OUT:

START

output no. of jobs and no. of mach.

output no. of operation for each job

output the p. t. dist. of "small" and "large" lot size mach

output the generated job-shop problem

RETURN
PROCEDURE  ADJUT :

START

- take the first job and its operation
- assign this job to available oper. job
- assign the first operation and its machine to available operation
- set the st. time for available op. to zero
- assign processing time to finish time available op.
- set the position to zero
- set the stage to one

RETURN
PROCEDURE INIT:

START

Initially set much next free to zero

RETURN
PROCEDURE ASSIGN-OPER-TO-MACH:

START

job equal 1

activate ASS-RANDOM procedure

chose at RANDOM one machine (1-mm)

chose at RANDOM one operation (1-op)

repeat until one operation is assigned to one machine

set flag to true

take first operation

C

D

has the machine randomly chosen assigned before

yes

set flag to false

no

chose at RANDOM another machine
CONTINUE PROCEDURE ASSIGN-OPER-TO-MACH:

D

has the oper. randomly chosen assigned before

yes

no

set flag to false

E

choose at random another oper.

F

is flag true

no

B

assign randomly chosen machine to randomly chosen operation

A

next operation

next job

RETURN
PROCEDURE SMALLEST-FINISH-TIME:

START

save the finish time of 1 available oper.

set index to 1

take the first available oper.

Does next available oper. has the larger finish time

yes put this available oper. into temp var.

no set the index to pos. of avail. oper.

next available oper. until finish

RETURN
PROCEDURE CHOSE-HU:

randomly choose N between 1-3.

is N=1

yes assign "RND" to heuristic

no

is N=2

yes assign "STP" to heuristic

no

is N=3

yes assign "FCFS" to heuristic

no

RETURN
PROCEDURE SCHE-OPER:

START!

set counter to zero

take the first aval. opera

is mach for this oper same as mach with earliest fin. ti.

yes increase counter

no note the aval. oper. on which this mach. is done

is this the last aval. opera

no assign that aval. oper. to partial schedule list

no note when this mach. is free next

is counter > 1

yes

no

RETURN
CONTINUE PROCEDURE SCHEL-OPER:

- Is heuristic RAND only?
  - Yes: RAND only chose one of available oper.
  - No: Set index to that sub.

- Is heuristic STP?
  - Yes: Call STP index
  - No: Call FCFS (index)

- Is heuristic FCFS?
PROCEDURE PREPARE-NXT-STAGE:

START

place the oper. no. of just sched. oper. in K

is K < no. of oper. allowed for this job?

yes

next oper. ready to be schedule is placed on ava. oper. list

no

set the job, machine operation, start time to negative and the finish time to infinite of the just sched. oper.

F

take the first mach

G

take the first job

for each job

the mach. used by this oper. is saved at ava. oper. list and start time and finish time

is the job same as just schedule?

yes

no
CONTINUE  PROCEDURE PRPAR-NXT-STAG :

- IS this mach same as mach set above? 
  - no: F
  - yes: continue

- IS this mach free? (st time < time this mach is free) 
  - no: G
  - yes: increase no of stages.

- IS oper. st. time > zero? 
  - no: H
  - yes: continue

st time of this oper is set to the time this mach is free

finish time is adjusted accordingly
CONTINUE  PROCEDURE  PRPAR-NEX-STAG:

1. Is this job last job?  
   - No \( \rightarrow \) F
   - Yes \( \rightarrow \) B

2. Is this last mach?  
   - No \( \rightarrow \) G
   - Yes \( \rightarrow \) take 1st mach

3. Take 1st mach  
   \( \rightarrow \) D

4. Take 1st job  
   \( \rightarrow \) B

5. Is this mach same as mach set above?  
   - No \( \rightarrow \) C
   - Yes \( \rightarrow \) E

6. Is oper st time > zero?  
   - No \( \rightarrow \) C
   - Yes \( \rightarrow \) W

7. Is this the last job?  
   - No \( \rightarrow \) E
   - Yes \( \rightarrow \) D

8. Is this the last mach?  
   - No \( \rightarrow \) E
   - Yes \( \rightarrow \) C

9. Finish time is adjusted accor.  
   \( \rightarrow \) C

10. Is time < time this mach is free?  
    - Yes \( \rightarrow \) end
    - No \( \rightarrow \) RETURN
PROCEDURE SORT:

```
START

take 1st scheduled job

is mach. no. for this job > mach. no. for next job
  yes
  swap the two jobs
  no
  is there any more jobs
    no
    take the 1st mach.
    initialize a counter
    while operations use this mach.
      yes
      increment counter
    no
    RETURN

  no
  take the 1st mach.
  no. of jobs using this mach. = counter
  is 1st time of this mach. > next
    yes
    swap the two jobs
    no
    is there any more mach.
      yes
      RETURN
```
PROCEDURE FIN-C-MAX :

START

set c-max to finish-time of first scheduled operation

take the second job

is the finish time for this > c-max?

yes

set c-max to this finish-time

no

is there any more job?

yes

no

RETURN
PROCEDURE FINAL SCHEDUL:

START

display the heading for the final schedule

display the heading for the job, oper., st. time, finish time

take the first machine

output all the jobs using this machine

is there any more machine

yes

no

output the c-max for this schedule

RETURN
VARIABLES:

JOB-SHOP: Array of records that hold the generated machine, operation, and processing time of all jobs.

TEMP: Temporary array of records which hold the generated machine, operation, and processing time of one job.

MACH-DIST: Two dimensional array that hold mean and standard deviation of small lot size machine.

OP: Array that hold number of operation for each job.

MCH-DIS-LARG: Two dimensional array that hold mean and standard deviation of large lot size machines.

HEURISTIC: Packed array that hold type of heuristic to be employed.

LASTR: Variable which hold the last random value computed.

LOT: Variable of character type holding the lot size of each set of jobs.

SCHEDULE: Is a text file used for an output external file.

SUM: Is product of number of machines and number of jobs.

I, J: Are counter which hold the number of times
to be repeated.

**FLAG**: Used to determine whether it is the first time going through IN-DATA.

**JJ**: Number of jobs.

**MM**: Number of machines.

**JM**: Product of JJ and MM.

**RATIO**: Hold job to machine ratio.

**MM-PLUS**: Number of machines.

**TEMP1PL**: Number of operation for one job.

**RND-MACH**: Machine number which is randomly generated.

**RND-OPER**: Operation number which is randomly generated.

**R1**: Random variable.

**INDEX**: Index of the machine.

**COUNTER**: Hold the value of large or small lot size.

**N**: Hold the randomly chosen lot size.

**MEAN**: Mean of the processing time of machine.

**ST-DEV**: Standard deviation of the processing time distribution of machine.

**PT**: Hold the generated processing time for any operation.

**OPER-AVAL**: Array of record which hold information about available operation such as job, machine, operation, start time, finish time, and position.

**PARTAL-SCHAL**: Array of records which hold the scheduled
operation.

T : Holds numbers of stages which jobshop has gone through.

INDEX : Index of operation with the smallest finished time.

MACH-NXT-FREE: Array that hold the time that this machine is free next.

NO-MACH : Hold number of machine.

PRO-TIME : Array that hold the processing time of available operation.

NO-STAGE : Array that hold the number of stage each available operation has been through.

C-MAX : Hold the completion time of the schedule.

COUNTER : Count the number of operation with smallest finish time that used the same machine.

RND-CHOOSE : Index for randomly chosen operation.

CHOOSE-ONE-OPER: Hold the randomly chosen operation.

TT : Temporary number of stage.

K : Position of the available operation.

X : Accumulated number of machine.

Y : counter to be checked for machine number.
PROGRAM GENERATE_JOB_SHOP (INPUT, OUTPUT, SCHEDULE);

*---------------------------------------------------------------------*

PROGRAM GENERATE_JOB_SHOP CONSISTS OF TWO PARTS:

1) GENERATION A JOB SHOP PROBLEM BY SIMULATION METHODS

2) SCHEDULE THIS JOB SHOP PROBLEM

INPUT:
- NUMBER OF MACHINES
- PROCESSING TIME DISTRIBUTION FOR LARGE AND SMALL LOTS

RANDOMLY GENERATED:
- JOB TO MACHINE RATIO
- PROCESSING TIME OF EACH OPERATION
- LOT SIZE
- TYPE OF HURISTIC TO BE USED

OUTPUT:
- GENERATED JOB SHOP PROBLEM
- FINAL SCHEDULE FOR JOB SHOP PROBLEM
- PERFORMANCE MEASURE (MAX FOR THIS SCHEDULE
- INDEPENDENT AND DEPENDENT VARIABLE FOR THE ANALYSIS

OBJECTIVE:
TO RUN THIS PROGRAM AS A SIMULATION PACKAGE FOR
A NUMBER OF TIMES THAT IS CALLED FOR BY THE EXPERIMENTAL
DESIGN. FOR EACH RUN FILE THE INDEPENDENT AND DEPENDENT
VARIABLE TO BE USED BY THE (SAS) FOR ITS ANALYSIS.

*---------------------------------------------------------------------*
CONST
  CONT = 10;
  SEED=0.1; (* for SEED < 1 *)

TYPE
  OP_MACH = RECORD
    MACHINE: OPERATION; INTEGER;
    PROCESSING_TIME: REAL;
  END;
  ONEJOB = ARRAY (. . . 100.) OF OP_MACH;

VAR
  JOB/shop: ARRAY (. . . 100) OF ONEJOB;
  GP: ARRAY (. . . 100) OF INTEGER;
  TEMP: ONEJOB;
  MACH_DIS: ARRAY (. . . 100 . . . 2) OF REAL;
  MCH_DIS_LARG: ARRAY (. . . 100 . . . 1 . . . 2) OF REAL;
  HURISTIC: PACHED ARRAY (. . . 4) OF CHAR;
  LOT: CHAR;
  LASTR: REAL; (* LAST RANDOM NO. *)
  SCHEDUL: TEXT;
  SUM,N,IRD,E,FLAG,HEU,0,00,10,RR,RR,RR,RR,RR,RR,RR,RR,RR,RR,RR,RR,RR : INTEGER;

FUNCTION RANDOM : REAL;

  CONST
    NF1 = 309.6;
    ADDER = 0.203125;

  BEGIN (* RANDOM *)
    LASTR := NF1 * LASTR + ADDER;
    LASTR := TRUNC(LASTR);
    RANDOM := LASTR;
  END; (* RANDOM *)

FUNCTION RANDINT: COUNTER : INTEGER ; INTEGER;
(--------------------------)

PROCEDURE RANDINT GENERATES A RANDOM INTEGER BETWEEN 1 AND COUNTER.
(--------------------------)
( * 1 <= RANDINT <= COUNTER *)

BEGIN (* RANDINT *)
  RANDINT I := 1 + TRUNC (COUNTER - RANDOM)
END; (* RANDINT *)
PROCEDURE IN DATA:

(* ----------------------------- *)

PROCEDURE IN DATA DOES THE FOLLOWING:

- INPUT NUMBER OF MACHINES
- INPUT PROCESSING TIME DISTRIBUTION FOR MACHINES OF
  SMALL AND LARGE LOT SIZE
- ASSIGN NUMBER OF OPERATION FOR EACH JOB
- FIND MACHINE/JOB RATIO
- DETERMINE NUMBER OF JOBS

(* ----------------------------- *)

VAR
I, J, COUNTER : INTEGER;
BEGIN (* I/O DATA *)
  IF (FLAG = C) THEN
    BEGIN
      WRITELN('Enter number of machines: ');
      READLN(MACH);
      FOR I := 1 TO MACH DO
        BEGIN
          WRITELN('Enter mean and st. dev. for process time of ');
          WRITELN('machine i: ');
          WRITELN('mean is: ');
          READLN(MACH DIS(I, 1));
          WRITELN('st. dev. is: ');
          READLN(MACH DIS(I, 2));
          WRITELN;
        END;
    END;
END:

END:

JJ := RATIO * MACH;
JM := JJ * MACH;
SUM := JM;
FOR J := 1 TO JJ DO
  OP(J, J) := MACH;
  FLAG := 1;
END; (* IN DATA *)
PROCEDURE INITIALIZE;

PROCEDURE INITIALIZE PREPARES THE JOB SHOP MATRIX BY SETTING ALL CELLS TO ZERO.

VAR
J, I: INTEGER;

BEGIN
FOR J := 1 TO JJ DO
BEGIN
TEMP := JOB SHOP(J, J);
FOR I := 1 TO OP(J, J) DO
BEGIN
TEMP(I, I).MACHINE := 0;
TEMP(I, I).OPERATION := 0;
TEMP(I, I).PROCESSING TIME := 0.0;
END;
JOB SHOP(J, J) := TEMP;
END;
END.
PROCEDURE ASSIGN OPER TO MACH:

PROCEDURE ASSIGN OPER TO MACH RANDOMERICAN OPERATION EACH OPERATION A MACHINE USING EQUALLY LIKELIHOOD CONCEPT TAKING INTO ACCOUNT THAT EACH JOB IS PROCESSED ON A MACHINE ONLY ONCE.

VAR
  J,K,TEMP1 : INTEGER;

PROCEDURE ASSIGN RANDOM VAR TEMP OPER.job ; TEMP1 (INTEGER):

VAR
  FLIGHT : BOOLEAN;
  I,MAPLUS,TEMP1,MAND,MARK,UENER : INTEGER;

BEGIN (* ASSIGN RANDOM *)

  MAPLUS := PLUS
  TEMP1 := TEMP;
  RND MACH := RNDINT (MAPLUS);
  RND OPER := RNDINT (TEMP1);
  REPEAT
    FLIGHT := TRUE;
    FOR I := 1 TO TEMPI DO
    BEGIN
      IF (TEMP1.I.).MACHINE = RND MACH THEN
      BEGIN
        FLIGHT := FALSE;
        RND MACH := RNDINT (MAPLUS);
      END;
      IF (TEMP1.I.).OPERATION = RND OPER THEN
      BEGIN
        FLIGHT := FALSE;
        RND OPER := RNDINT (TEMP1);
      END;
    END;
    UNTIL FLIGHT;
  TEMP1.I.).MACHINE := RND MACH;
  TEMP1.I.).OPERATION := RND OPER;
  END; (* ASSIGN RANDOM *)

BEGIN (* ASSIGN OPER TO MACH *)

  FOR J := 1 TO JU DO
  FOR I := 1 TO OPER(J).DO
  BEGIN
    TEMP1 := OPER(J);
    TEMP := JOB SHIP(I,J);
    ASSIGN RANDOM (TEMP,TEMP1);
    JOB SHIP(J,J) := TEMP;
  END;
  END; (* ASSIGN OPER TO MACH *)
FUNCTION ST NORMAL : REAL:

CONST
  PI = 3.14:
VAR
  K,R1 : REAL:
BEGIN (* ST NORMAL *)
  K := SORT(S/PI);
  R1 := RANDOM:
  ST_NORMAL := (1/2) * LRM(1+R1/(1-R1)):
END; (* ST NORMAL *)

PROCEDURE PROCES_TIME:

(*/

PROCEDURE PROCESS_TIME DOES THE FOLLOWING:
- DETERMINES THE LOT SIZE OF EACH JOB
- DEPENDS ON THE LOT SIZE PROCESSING TIME FOR EACH JOB:
- THE LOT SIZE IS CALCULATED USING PROCESS TIME DISTRIBUTION AND STANDARD NORMAL PROBABILITY.*/

VAR
  J,I,INDEX,COUNTER,N : INTEGER;
  MEAN,ST_DEV,PT : REAL:
BEGIN (* PROCES_TIME *)
  COUNTER := 2;
  FOR J := 1 TO JJ DO
    BEGIN
      TEMP := JOB SHOP(J,.)
      FOR I := 1 TO OPT(J,.) DO
        BEGIN
          IF (BATCH = 1) THEN
            BEGIN
              INDEX := TEMP(I,).MACHINE:
              MEAN := MACH DIST (INDEX, 1):
              ST_DEV := MACH DIST (INDEX, 2):
              PT := ST_DEV + ST_NORMAL + MEAN:
              TEMP(I,).PROCESSING_TIME := PT;
              LOT := 'S';
            END
          ELSE IF (BATCH = 2) THEN
            BEGIN
              INDEX := TEMP(I,).MACHINE:
              MEAN := MACH DIST (INDEX, 1):
              ST_DEV := MACH DIST (INDEX, 2):
              PT := ST_DEV + ST_NORMAL + MEAN:
              TEMP(I,).PROCESSING_TIME := PT;
              LOT := 'L';
            END
          END
        END
      END;
    END;
  END; (* PROCES_TIME *)
PROCEDURE SOR.OP:

PROCEDURE SOR.OP TAKES THE RANDOMLY GENERATED JOE
SHOP AND SORT EACH JOB ACCORDING TO ITS OPERATIONS. WE
RESULT THE MACHINE AND PROCESSING TIME IS ADJUSTED. THIS
GIVES A JOB SHOP PROBLEM WHICH ITS OPERATIONS FOR EACH JOB
IS IN INCREASING ORDER.

VAR

I, J : INTEGER;

T1 : OF_MACH;

BEGIN (* SOR.OP *)

FOR J := 1 TO JJ DO

BEGIN

TEMP := JOE.SHOP(J, J);

FOR I := 1 TO OP(J, I) - 1 DO

BEGIN

FOR W := 1 TO OP(J, I) - 1 DO

BEGIN

IF (TEMP(W, I).OPERATION) >

TEMP(W - 1, I).OPERATION) THEN

BEGIN

T1 := TEMP(W, I);

TEMP(W, I) := TEMP(W - 1, I);

TEMP(W - 1, I) := T1;

END:

END;

END;

JOE.SHOP(J, I) := TEMP;

END;

END (* SOR.OP *)
PROCEDURE PRINT OUT:
(**************************************************************************

PROCEDURE PRINT OUT PRODUCES A HARD COPY OF THE
GIVEN INPUTS TO THE SYSTEM SUCH AS NO. OF MACHINES AND
PROCESSING TIME DISTRIBUTION FOR EACH MACHINE AND LOT SIZE.
ALSO, IT GIVES THE RANDOMLY GENERATED JOB SHOP PROBLEM.
**************************************************************************)

VAR
I,J: INTEGER;
BEGIN (* PRINT OUT *)

WRITELN;
WRITELN;
WRITELN('GIVEN INPUT ***');
WRITELN;
WRITELN('NUMBER OF JOBS IS : ', I);  
WRITELN('NUMBER OF MACHINES IS : ', J);
WRITELN;
WRITELN('NUMBER OF OPERATION FOR EACH JOB ***');
WRITELN;
WRITELN('JOB NO. NO. OF OPERATION ***');
WRITELN;
FOR J := 1 TO J DO
WRITELN(J);  
WRITELN(J);  
WRITELN;
WRITELN('MACH TIME DISTR FOR SMALL LOTS ** **');
WRITELN;
WRITELN('MACH NO. MEAN ST DEV');
WRITELN('------------------ ------ ----');
FOR I := 1 TO I DO
WRITELN(I, I : 1, 1 12, I, 12);  
WRITELN;
WRITELN('MACH TIME DISTR FOR LARGE LOTS ** ');
WRITELN;
WRITELN('MACH NO. MEAN ST DEV');
WRITELN('------------------ ------ ----');
FOR I := 1 TO I DO
WRITELN(I, I : 1, 1 12, I, 12);  
WRITELN;
WRITELN;
WRITELN;
WRITELN;
WRITELN('TEMP : JOB SHOP : JOB : ');
WRITELN('OPERATION : MACHINE : PROCESSING TIME');
WRITELN('------------------ ------ --------------');
FOR I := 1 TO I DO
BEGIN
WRITELN(TEMP(I), I : 1, 1 12);  
WRITELN(Operation : I, I 12);  
WRITELN(MACHINE : I 12);  
WRITELN(Processing Time : I 12);  
END;
END;
WRITELN;
WRITELN;
WRITELN;
END; (* OUT_PUT *)
PROCEDURE SCHUL:
(*)

PROCEDURE SCHUL USES SCHEDULE ALGORITHM TECHNIQUES
AND TYPE OF HURISTIC, RANDOMLY CHOSEN, TO SCHEDULE THE JOB
SHOP PROBLEM.

(******************************************************************************)

TYPE
REC. TYPE = RECORD
  JOB, MACH OPERATION : INTEGER;
  START TIME, FINISH TIME : REAL;
  POS : INTEGER;
END;
PAR. SCH = RECORD
  JOB, MACH OPERATION : INTEGER;
  START TIME, FINISH TIME : REAL;
END;

VAR
OPRY AVGHL : ARRAY (1..100) OF REC. TYPE;
PARTIAL SCHUL = ARRAY (1..100) OF PAR. SCH;
T, INDEX : INTEGER;
MACH, NXT FREE : ARRAY (1..100) OF REAL;
NOD MACH : ARRAY (1..100) OF INTEGER;
PRO, TIME : ARRAY (1..100) OF REAL;
NO STAGE : ARRAY (1..100) OF INTEGER;
C. MAX : REAL;

PROCEDURE INIT:
(*)

PROCEDURE INIT, INITIALIZES THE ARRAY WHICH HOLDS
THE INFORMATION ON WHEN A PARTICULAR MACHINE IS NEXT FREE
AFTER BEING ASSIGNED TO A OPERATION.

(******************************************************************************)

VAR
  I : INTEGER;
BEGIN
  I := 1 TO 100 DO;
  MACH, NXT FREE (1..,1) := 0.0;
END;

(*)
PROCEDURE ADJUT:

( * This procedure calculates the earliest finish time for each job. Initially, all the jobs are available to be scheduled. So, the first operation of each job within its specifications is moved to section available operation and stage is set to one. *
)

VAR
J : INTEGER;
BEGIN (* ADJUT *)
FOR J := 1 TO JJ DO
BEGIN
  TEMP := JOB SHOP (.J,.J).END;
  OPER AVAIL (.J,.J).START TIME := 0;
  PROF TIME (.J,.J) := TEMP (.J,.J).PROCESSING TIME;
  OPER AVAIL (.J,.J).POS := 1;
  NO STAGE (.J,.J) := 1;
END;
INIT;
END; (* ADJUT *)

PROCEDURE SMALLEST_FINISH_TIME:

( * This procedure searches through the available operation and finds the one operation that has the earliest finish time. Then note the job on which that operation is belonging to. *
)

VAR
TEMP := REAL;
I := INTEGER;
BEGIN (* SMALLEST_FINISH_TIME *)
BEGIN
  TEMP := OPER AVAIL (.I,.I).FINISH TIME;
  INDEX := I;
  FOR I := 1 TO JJ DO
  BEGIN
    IF (OPER AVAIL (.I,.I).FINISH TIME < TEMP) THEN
    BEGIN
      TEMP := OPER AVAIL (.I,.I).FINISH TIME;
      INDEX := I;
    END;
  END;
END; (* SMALLEST_FINISH_TIME *)
PROCEDURE SCHU OPEN;

(* =========================================================================

PROCEDURE SCHU OPEN SEARCHES THROUGH ALL THE AVAILABLE OPERATIONS AND
COMPARES THE MACHINE, USED BY THE OPERATION WITH THE EARLIEST FINISH TIME,
TO MACHINES USED BY REST OF THE AVAILABLE OPERATIONS.

IF THERE IS ANY OTHER OPERATION WHICH USES THE SAME MACHINE, THE JOB ON
WHICH THIS OPERATION BELONGS TO AND NO. OF JOBS THAT USED THE SAME MACHINE ARE SAVED FOR LATER USE.

IF THERE ARE MORE THAN ONE AVAILABLE OPERATION USING SAME MACHINE, ONE IS CHOSEN ACCORDING TO THE HURISTIC.

FINALLY, THAT PARTICULAR OPERATION SCHEDULED ON THAT MACHINE AND NEXT TIME THIS MACHINE IS FREE IS CALCULATED AND SAVED.

============================================================================)

VAR
OUNTER,1,CHOSE_1 : INTEGER;
CHOSE_ONE OPER : ARRAY (.1.100.) OF INTEGER;

PROCEDURE SPT(VAR INDEX : INTEGER);

(* =========================================================================

PROCEDURE SPT IS ONE OF THE HURISTIC USED BY THE SCHEDULING ALGORITHM. THIS HURISTIC IS BASED ON SHORTEST PROCESSING TIME.

============================================================================)

VAR
I : INTEGER;
TT : REAL;
BEGIN (* SPT *)

TT := PRO TIME(.CHOSE ONE OPER(.1.));
INDEX := CHOOSE_1 OPER(.1.);
FOR I := 2 TO COUNTER DO
BEGIN
IF (PRO TIME(.CHOSE ONE OPER(.1.)) < TT) THEN
BEGIN
TT := PRO TIME(.CHOSE ONE OPER(.1.));
INDEX := CHOOSE_1 OPER(.1.);
END;
END (* SPT *)
PROCEDURE FCFS(VAR INDEX : INTEGER);
(*=================================================================

PROCEDURE FCFS IS ONE OF THE HURISTIC USED IN THE
SCHEDULING ALGORITHM. THIS HURISTIC IS USED ON FIRST COME
FIRST SERVED. THIS PROCEDURE IS TO FIND WHICH AVAILABLE
OPERATION HAS BEEN HAVING IN THE QUEUE THE LONGEST TIME.

=================================================================* )

VAR
I, IT : INTEGER;
BEGIN (* FCFS *)

IT := NO_STAGE(.CHOOSE ONE OPER(.1.));
INDEX := CHOOSE ONE OPER(.1.);
FOR I := 2 TO COUNTER DO
BEGIN
IF (NO_STAGE(.CHOOSE ONE OPER(.1.)) > IT) THEN
BEGIN
IT := NO_STAGE(.CHOOSE ONE OPER(.1.));
INDEX := CHOOSE ONE OPER(.1.);
END;
END;
END; (* FCFS *)

BEGIN (* SCHU OPER *)
COUNTER := 0;
FOR I := 1 TO JJ DO
BEGIN
IF (OPER AVAIL(.1.) .MACH = OPER AVAIL(.INDEX,) .MACH) THEN
BEGIN
COUNTER := COUNTER + 1;
CHOOSE ONE OPER(.COUNTER.) := 1;
END;
END;
IF (CHEU = 1) THEN
HURISTIC := "FIFO"
ELSE IF (CHEU = 2) THEN
HURISTIC := "SPT"
ELSE IF (CHEU = 3) THEN
HURISTIC := "FIFO";
IF (COUNTER > 1) THEN
BEGIN
IF (HURISTIC = "FIFO") THEN
BEGIN
AND CHOOSE := END INT(COUNTER);
INDEX := CHOOSE ONE OPER(.AND CHOOSE.);
END;
IF (HURISTIC = "SPT") THEN
SPT(INDEX);
IF (HURISTIC = "FCFS") THEN
FCFS(INDEX);
END:
PARTIAL SCHULL(T), .MACH := OPER AVAIL(.INDEX,) .MACH;
PARTIAL SCHULL(T), .OPERATION := OPER AVAIL(.INDEX,) .OPERATION;
PARTIAL SCHULL(T), .START TIME := OPER AVAIL(.INDEX,) .START TIME;
PARTIAL SCHULL(T), .FINISH TIME := OPER AVAIL(.INDEX,) .FINISH TIME;
MACH .NOT FREE := OPER AVAIL(.INDEX,) .MACH;
END; (* SCHU OPER *)
PROCEDURE PRFAR NXT STAG;
BEGIN

PROCEDURE PRFAR NXT STAG LOOKS AT THE OPERATION JUST SCHEDULED AND CHECKS TO SEE IF ANY MORE OPERATIONS ARE LEFT FROM THAT JOB. IF SO, THE NEXT OPERATION FOR THAT JOB IS SAVED AS AVAILABLE OPERATION ALONG WITH ALL ITS COMPONENTS: MACHINE BEING USED ON, START TIME, FINISH TIME, PROCESSING TIME, AND THE POSITION. OTHERWISE, THAT JOB IS ELIMINATED FROM THE LIST.

ALSO, ALL THE AVAILABLE OPERATIONS THAT USE MACHINE THAT JUST SCHEDULED RECEIVE AN UPDATED START TIME.

VAR
K, I, J : INTEGER;
TEMP : ONE_JOB;

BEGIN (* PRFAR NXT STAG *)

K := OPER.AVAL(INDEX,.POS);
IF (I < CH(INDEX,)) THEN
BEGIN
 TEMP := JOB(SHOP(INDEX,);
 OPER.AVAL(INDEX,.MACH := TEMP,.K+1,.MACHINE;
 OPER.AVAL(INDEX,.OPERATION := TEMP,.K+1,.OPERATION;
 OPER.AVAL(INDEX,.POS := I+1;
 OPER.AVAL(INDEX,.START_TIME := OPER.AVAL(INDEX,.START_TIME;
 + TEMP,.K+1,.PROCESSING_TIME;
 OPER_AVAL(INDEX,.FINISH_TIME := OPER_AVAL(INDEX,.START_TIME;
 + TEMP,.K+1,.PROCESSING_TIME;
 NO_STAGE(INDEX, := TEMP,.K+1,.PROCESSING_TIME;
 FOR I := 1 TO J DO
 IF (I < INDEX) THEN
 IF (NO_STAGE(INDEX, := 0) THEN

NO STAGE(I, J) := NO STAGE(I, J) + 1;
FOR I := 1 TO M DO
FOR J := 1 TO JJ DO
BEGIN
TEMPJ := JOBショップ(J, J);
IF (OPER AVAILABLE(J, J), MACHINE = I) THEN
(OPER AVAILABLE(J, J), START TIME := 0) THEN
BEGIN
IF (OPER AVAILABLE(J, J), START TIME < MACHINE NOT FREE(I, J))
THEN
BEGIN
OPER AVAILABLE(J, J), START TIME := MACHINE NOT FREE(I, J);
OPER AVAILABLE(J, J), FINISH TIME := OPER AVAILABLE(J, J), START TIME
+ TEMP(J, OPER AVAILABLE(J, J), POS), PROCESSING TIME;
END;
END;
END;
ELSE
BEGIN
OPER AVAILABLE(J, INDEX), JOB := -1;
OPER AVAILABLE(J, INDEX), MACHINE := I;
OPER AVAILABLE(J, INDEX), OPERATION := -1;
OPER AVAILABLE(J, INDEX), START TIME := -1;
OPER AVAILABLE(J, INDEX), FINISH TIME := 2000000;
Mach Free(J, INDEX) := 1;
OPER AVAILABLE(J, INDEX), POS := OPER(J, INDEX) + 1;
FOR IT := 1 TO JJ DO
FOR J := 1 TO JJ DO
BEGIN
TEMPJ := JOB SHOP(J, J);
IF (OPER AVAILABLE(J, J), MACHINE = I) THEN
IF (OPER AVAILABLE(J, J), START TIME = 0) THEN
BEGIN
IF (OPER AVAILABLE(J, J), START TIME < MACHINE NOT FREE(I, J))
THEN
BEGIN
OPER AVAILABLE(J, J), START TIME := MACHINE NOT FREE(I, J);
OPER AVAILABLE(J, J), FINISH TIME := OPER AVAILABLE(J, J), START TIME
+ TEMP(J, OPER AVAILABLE(J, J), POS), PROCESSING TIME;
END;
END;
END;
END;
T := T + 1;
END;
PROCEDURE SORT:

(*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

PROCEDURE SORT TAKES THE SCHEDULE JOB SHIP PROBLEM AND
SORT THEM IN ORDER ON MACHINES.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*)

VAR

I,J,W,X,Y : INTEGER;
Ti : Partial_SCHUL:i;

BEGIN (* SORT *)
FOR I := 1 TO SUM-1 DO
BEGIN
    FOR J := 1 TO SUM-1 DO
    BEGIN
        IF (PARTIAL_SCHUL(J).MACH > PARTIAL_SCHUL(J+1).MACH) THEN
        BEGIN
            Ti := PARTIAL_SCHUL(J);
            PARTIAL_SCHUL(J) := PARTIAL_SCHUL(J+1);
            PARTIAL_SCHUL(J+1) := Ti;
        END;
    END;
END;
Y := 1;
PARTIAL_SCHUL(SUM+1).MACH := 0;
FOR I := 1 TO NUM DO
BEGIN
    X := 0;
    WHILE (PARTIAL_SCHUL(Y).MACH = 1) DO
    BEGIN
        X := X + 1;
        Y := Y + 1;
    END;
    NUM_MACH(I) := X;
END;
X := 0;
FOR W := 1 TO NUM DO
BEGIN
    IF W = NUM_MACH(W) - 1 DO
    BEGIN
        FOR J := 1 TO NUM_MACH(W) - 1 DO
        BEGIN
            IF (PARTIAL_SCHUL(J+X).START_TIME > PARTIAL_SCHUL(J+X+1).START_TIME) THEN
            BEGIN
                Ti := PARTIAL_SCHUL(J+X);
                PARTIAL_SCHUL(J+X) := PARTIAL_SCHUL(J+X+1);
                PARTIAL_SCHUL(J+X+1) := Ti;
            END;
        END;
    END;
END;
X := X + NUM_MACH(W);*

END (* SORT *)
PROCEDURE FINAL_SCHUL;
(* **********************************************************************/

PROCEDURE FINAL_SCHUL PRINTS THE SCHEDULE IN ORDER OF MACHINES. FOR EACH MACHINE THE JOB NO., OPERATION, START TIME AND FINISH TIME IS PRINTED. ALSO, THE COMPLETION TIME CALCULATED FOR THIS SCHEDULE IS PRINTED.

******************************************************************************/

VAR
    I, J, X : INTEGER;
BEGIEN (* FINAL_SCHUL *)
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
   WRITELN;
   WRITELN;
   WRITELN;
   X := 0;
   FOR J := 1 TO NO_MACH(J) DO
     BEGIN
       WRITELN(" FOR MACHINE ", J, ", ");
       WRITELN;
       WRITELN(" JOB OPERATION START TIME FINISH TIME ");
       WRITELN;
       FOR I := 1 TO NO_MACH(J) DO
         BEGIN
           WRITE(PARTIAL_SCHUL(I+X,J));
           WRITE(PARTIAL_SCHUL(I+X,J),OPERATION :12);
           WRITE(PARTIAL_SCHUL(I+X,J),START TIME :15:12);
           WRITE(PARTIAL_SCHUL(I+X,J),FINISH TIME :15:12);
         END;
         X := X + NO_MACH(J);
         WRITELN;
       END;
       WRITELN;
       WRITELN(" ------------------------------- ");
       WRITELN;
       WRITELN;
       WRITELN(" C_MAX = ", C_MAX:12);
       WRITELN(" ------------------------------- ");
       WRITELN;
     END; (* FINAL_SCHUL *)
PROCEDURE FIND C MAX:
(* THIS IS THE PERFORMANCE MEASURE. COMPLETION TIME IS THE EARLIEST TIME THAT ALL THE JOBS ARE COMPLETED. *)

VAR
I : INTEGER;
BEGIN (* FIND C MAX *)
C_MAX := PARTIAL_SCHUL(I).FINISH_TIME;
FOR I := 1 TO SUM DO
  IF (PARTIAL_SCHUL(I).FINISH_TIME < C_MAX) THEN
    C_MAX := PARTIAL_SCHUL(I).FINISH_TIME;
END; (* FIND C MAX *)

PROCEDURE PRINT SAS DATA:
(* THIS PRINTS THE INFORMATION NEEDED BY THE SAS SUCH AS RATIO, LOT SIZE, HURISTIC, AND C MAX TO AN EXTERNAL FILE. *)

BEGIN (* PRINT SAS DATA *)
  WRITELN(SCHEDULE.RATIO:2,LOT:4,HURISTIC:4,C_MAX:12:2);
END; (* PRINT SAS DATA *)

BEGIN (* MAIN *)
  ADJU;
  T:=1;
  WHILE (T <= (J*M)) DO
    BEGIN
      SMALLEST FINISH TIME;
      SCHU OFFER;
      PREPARE NAT S160;
    END;
    SMALLEST;
    FIND C_MAX;
    FINAL SCHUL;
    PRINT, SAS DATA;
END; (* MAIN *)
BEGIN (* MAIN *)

LASTR := SEED;
REWRITE(SCHEDULE);
FLAG := 0;
WRITELN('------ HOW MANY SCHULING DO YOU NEED ? ');
READLN(N);
WRITELN;
FOR III := 1 TO N DO
BEGIN
FOR I:= 1 TO 10 DO
  FOR JJ := 1 TO 3 DO
    FOR K := 1 TO 2 DO
      BEGIN
        RATIO := 1;
        HEL := JJ;
        RATIO := 1;

        IN_DATA;
        INITIALIZE;
        ASSIGN_OPER TO MACH;
        PROCES_TIME;
        SCR_OPR;

        PRINT OUT;
        SCHUL;
      END;
END;
END. (* MAIN *)
END.