HEURISTIC SEQUENCING OF SINGLE AND MULTIPLE
COMPONENT JOBS

by

DONALD CARY CARROLL

B.S., University of North Carolina, 1954
S.M., Massachusetts Institute of Technology, 1958

Submitted in Partial Fulfillment
of the Requirements for the Degree
of Doctor of Philosophy in
Industrial Management

June 1965

Signature of Author

Alfred P. Sloan School of Management, May 15, 1965

Certified by

Thesis Supervisor

Accepted by

Chairman, Departmental Committee on Graduate Students

"Work reported herein was supported (in part)
by Project MAC, an M.I.T. research program
sponsored by the Advanced Research Projects
Agency, Department of Defense, under Office of
Naval Research Contract Number Nonr-4102(01).
Reproduction in whole or in part is permitted for
any purpose of the United States Government."
HEURISTIC SEQUENCING OF
SINGLE AND MULTIPLE
COMPONENT JOBS

by

DONALD CARY CARROLL

Submitted to the Alfred P. Sloan School of Management on May 15, 1965
in partial fulfillment of the requirements for the degree of Doctor
of Philosophy in Industrial Management.

ABSTRACT

The purposes of this research are to extend understanding of more
realistic versions of the job shop sequencing problem, to propose and
test heuristics designed to cope with some of the new complications, and
to evaluate access to a "global" data base in real-time, this being a
requirement to support the more elaborate heuristics. The additional
features of the job shop studied here are multiple channel work stations
and multiple component orders, i.e., orders involving assembly operations.
The former introduces channel availability time as relevant data for
sequencing, the latter, measurements of relative progress of components
(something similar to "critical path" information).

The basic model is similar to those used previously. It assumes,
among other things, known processing times, nearly "pure" job shop
routings, continuous manning of work stations, and sequence-independent
setup costs. Exponential arrivals and service times are assumed for the
most part; order release times and due dates are exogenously supplied.
A departure is made from previous practice by assuming that early (i.e.,
prior to due date) shipments are not permitted and that all orders incur
tardiness penalties at the same (linear) rate. Since all sequencing
systems are submitted to the same inputs, a single measure of effective-
ness, namely, mean order tardiness, is thereby established.

The sequencing system proposed here, called COVERT (for "c over t"),
is based on a theorem by Wayne Smith (and others) to the effect that to
sequence n jobs through a single machine, given that delay cost rates $c_i$
and processing times $t_i$ are known, the minimum delay cost is obtained by
sequencing by decreasing $c_i/t_i$ ratio. For a dynamic system, it is
necessary to approximate to obtain \( c \). Also, it is useful to consider arriving jobs as well as those present, channel availability times for multiple channels, and component "slack" for multiple components. Heuristics based on the imputed delay costs are developed for these problems.

The experimental medium is a digital simulation program developed by the author. Considerable effort is taken to rationalize the solutions to "tactical problems" noted by Conway as inherent in simulation experimentation.

The experiments (semi-factorial in load level, due-date tightness, shop configuration, and order type) lead to the following conclusions. The basic COVERT rules are highly effective relative to previously proposed rules under all conditions tested. The complex heuristic modifications of COVERT are effective but not of an overwhelming magnitude; the modifications for the multiple channel case are uninteresting. Significant value for the current, global data base was not demonstrated.

Thesis Supervisor: Edward H. Bowman
Title: Professor of Industrial Management
May 15, 1965

Professor William C. Green
Secretary of the Faculty
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Dear Professor Greene:

In accordance with the requirements for graduation, I herewith submit a thesis entitled "Heuristic Sequencing of Single and Multiple Component Jobs."

I would like to express my gratitude to my thesis supervisor, Professor Edward H. Bowman, for his advice, encouragement and prodding. Also, I would like to thank my committee members, Professors Martin Greenberger, Geoffrey Clarkson and William Pounds for several helpful suggestions. My longtime colleague, Professor James Emery, also contributed many thoughts.

Anyone who performs research in digital simulation, especially in job shop sequencing is indebted to Professors Richard Conway and William Maxwell of Cornell University. They have established standards both of rigor and imagination which are inspirational. I hereby acknowledge my debt to them.

Special thanks are due to Mrs. Janet Kinasewich, Mrs. Elizabeth Schneider, and Miss Carole Robinson who typed drafts and final copy of the thesis. Their perserverance in coping with a writer incapable of distinguishing among u's, n's, m's, and w's (among others) in handwritten copy was highly to be commended.

Finally, my thanks are due to my wife and children who were forced to subsist on less than their rightful share of my attention for a year.

Yours sincerely,

Signature redacted

Donald Cary Carroll
Restricted Distribution

This thesis has been given limited reproduction with the permission of the Alfred P. Sloan School of Management. However, this copy is issued with the understanding that none of the data resulting from this investigation will be used for advertising or publicity purposes, and that the thesis is solely for the confidential use of the organization or individual to whom it is addressed.
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>2</td>
</tr>
<tr>
<td>LETTER OF TRANSMITTAL</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>9</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>10</td>
</tr>
<tr>
<td>CHAPTER I. INTRODUCTION</td>
<td>11</td>
</tr>
<tr>
<td>Preamble</td>
<td>11</td>
</tr>
<tr>
<td>On-Line, Real-Time Systems and Incomplete Information</td>
<td>14</td>
</tr>
<tr>
<td>What is Known: A Thumbnail Assessment</td>
<td>16</td>
</tr>
<tr>
<td>Holt's Proposal</td>
<td>20</td>
</tr>
<tr>
<td>Delineation of Problem and Model</td>
<td>20</td>
</tr>
<tr>
<td>Some Definitions</td>
<td>22</td>
</tr>
<tr>
<td>Outline of Material which Follows</td>
<td>24</td>
</tr>
<tr>
<td>CHAPTER II. PROBLEM SETTING AND PREVIOUS RESEARCH</td>
<td>26</td>
</tr>
<tr>
<td>The Job Shop Environment</td>
<td>26</td>
</tr>
<tr>
<td>The Planning, Scheduling, and Sequencing Hierarchy</td>
<td>27</td>
</tr>
<tr>
<td>Economic Criteria Problems</td>
<td>30</td>
</tr>
<tr>
<td>Approaches to Research</td>
<td>32</td>
</tr>
<tr>
<td>Common Assumptions</td>
<td>36</td>
</tr>
<tr>
<td>The Literature</td>
<td>38</td>
</tr>
<tr>
<td>Static Microanalysis: The N-job Problem</td>
<td>38</td>
</tr>
<tr>
<td>Multiple Stages and Multiple Channels</td>
<td>41</td>
</tr>
<tr>
<td>Dynamic Microanalysis: Queuing Studies</td>
<td>43</td>
</tr>
<tr>
<td>Static Queue Networks</td>
<td>44</td>
</tr>
<tr>
<td>Dynamic Macroanalysis</td>
<td>47</td>
</tr>
<tr>
<td>Holt's Proposal</td>
<td>57</td>
</tr>
<tr>
<td>CHAPTER III. TOWARDS A RATIONALE FOR SEQUENCING</td>
<td>58</td>
</tr>
<tr>
<td>Introduction and Summary</td>
<td>58</td>
</tr>
<tr>
<td>Some Notation</td>
<td>58</td>
</tr>
<tr>
<td>Some Operating Characteristics of Rules</td>
<td>60</td>
</tr>
<tr>
<td>The Effect of Variable Flow Allowances</td>
<td>66</td>
</tr>
<tr>
<td>Variance Reducing Rules</td>
<td>70</td>
</tr>
<tr>
<td>Variance Reducing Heuristics</td>
<td>71</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (continued)

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The C/T Rule: &quot;COVERT!&quot;</td>
<td>74</td>
</tr>
<tr>
<td>Some Advantages of the COVERT Rule</td>
<td>79</td>
</tr>
<tr>
<td>Some Disadvantages of the COVERT Rule</td>
<td>80</td>
</tr>
<tr>
<td>Gere's &quot;Look-Ahead&quot; and &quot;Insert&quot;</td>
<td>81</td>
</tr>
<tr>
<td>A Basis for &quot;Hold-off&quot; and &quot;Sneak-In&quot;</td>
<td>82</td>
</tr>
<tr>
<td>Operating Characteristics</td>
<td>84</td>
</tr>
<tr>
<td>Multiple Channel Work Stations</td>
<td>86</td>
</tr>
<tr>
<td>Hold-off Rule for Multiple Channels</td>
<td>87</td>
</tr>
<tr>
<td>Heuristics for Serial-Parallel Routings</td>
<td>88</td>
</tr>
<tr>
<td>Serial-Parallel Orders: Formal Description</td>
<td>89</td>
</tr>
<tr>
<td>Schedules and Computation of Slack</td>
<td>94</td>
</tr>
<tr>
<td>The Significance of Segment Slack</td>
<td>96</td>
</tr>
<tr>
<td>General Scheme for Sequencing</td>
<td>97</td>
</tr>
<tr>
<td>Hold-off Rules</td>
<td>97</td>
</tr>
<tr>
<td>Single Channel Work Stations</td>
<td>98</td>
</tr>
<tr>
<td>Multiple Channel Machine Groups</td>
<td>98</td>
</tr>
<tr>
<td><strong>CHAPTER IV. STATEMENT OF HYPOTHESES AND EXPERIMENTAL PLAN.</strong></td>
<td>100</td>
</tr>
<tr>
<td>Introduction and Summary</td>
<td>100</td>
</tr>
<tr>
<td>Rules to be Tested</td>
<td>100</td>
</tr>
<tr>
<td>Statement of Hypotheses</td>
<td>102</td>
</tr>
<tr>
<td>The Simulation Program</td>
<td>105</td>
</tr>
<tr>
<td>Organization of the Program</td>
<td>106</td>
</tr>
<tr>
<td>A. The Input Generator</td>
<td>107</td>
</tr>
<tr>
<td>B. The Main Program</td>
<td>108</td>
</tr>
<tr>
<td>Some Special Programming Features</td>
<td>112</td>
</tr>
<tr>
<td>The Rules Themselves</td>
<td>113</td>
</tr>
<tr>
<td>Dynamic Critical Path Analysis</td>
<td>113</td>
</tr>
<tr>
<td>Experimental Design</td>
<td>115</td>
</tr>
<tr>
<td>Tactical Plan: Order Generation</td>
<td>115</td>
</tr>
<tr>
<td>The Tactical Problems</td>
<td>118</td>
</tr>
<tr>
<td><strong>CHAPTER V. RESULTS.</strong></td>
<td>124</td>
</tr>
<tr>
<td>Introduction</td>
<td>124</td>
</tr>
<tr>
<td>Sequencing Single Component Orders</td>
<td>124</td>
</tr>
<tr>
<td>Sensitivity Analysis: Load Effects</td>
<td>128</td>
</tr>
<tr>
<td>Sensitivity Analysis: Flow Allowance Effects</td>
<td>129</td>
</tr>
<tr>
<td>Sensitivity Analysis: Setting the COVERT &quot;K&quot;</td>
<td>130</td>
</tr>
<tr>
<td>Evaluation of Hold-off and Sneak-In (WORKS)</td>
<td>132</td>
</tr>
<tr>
<td>The Measurement Problem</td>
<td>133</td>
</tr>
<tr>
<td>Some Limited Conclusions</td>
<td>135</td>
</tr>
<tr>
<td>Single Component Orders with Multiple Channel Machine Groups</td>
<td>135</td>
</tr>
<tr>
<td>The General Shop</td>
<td>137</td>
</tr>
<tr>
<td>Sequencing Multiple-Component Orders</td>
<td>138</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (continued)

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER VI. SUMMARY AND CONCLUSIONS</td>
<td>145</td>
</tr>
<tr>
<td>Introduction</td>
<td>145</td>
</tr>
<tr>
<td>Summary of Research</td>
<td>145</td>
</tr>
<tr>
<td>Implications for Job Shop Sequencing</td>
<td>146</td>
</tr>
<tr>
<td>Implications for Heuristic Programming</td>
<td>149</td>
</tr>
<tr>
<td>On-Line, Real-Time Systems</td>
<td>150</td>
</tr>
<tr>
<td>The Model and Experimental Method</td>
<td>150</td>
</tr>
<tr>
<td>Collateral Research and Further Study</td>
<td>151</td>
</tr>
<tr>
<td>APPENDIX A. THE INFLUENCE OF MEAN AND VARIANCE ON NORMAL QUADRATIC TARDINESS LOSS FUNCTIONS</td>
<td>153</td>
</tr>
<tr>
<td>APPENDIX B. DESCRIPTION OF SELECTED HEURISTICS</td>
<td>157</td>
</tr>
<tr>
<td>APPENDIX C. LOGIC FOR UPDATING SEGMENT SLACK</td>
<td>171</td>
</tr>
<tr>
<td>APPENDIX D. SUMMARY OF EXPERIMENTAL RESULTS</td>
<td>176</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>180</td>
</tr>
<tr>
<td>BIOGRAPHICAL NOTE ON THE AUTHOR</td>
<td>185</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Number of Tasks per Order for Various Order Types</td>
<td>116</td>
</tr>
<tr>
<td>5.1</td>
<td>Mean Tardiness per Order, Shop 1, Order Type 1, U = .80</td>
<td>125</td>
</tr>
<tr>
<td>5.2</td>
<td>Mean Wait Times, Shop 1, Order Type 1, U = .80</td>
<td>126</td>
</tr>
<tr>
<td>5.3</td>
<td>Mean Tardiness per Order, Shop 1, Order Type 1, U = .7, .8, and .9</td>
<td>129</td>
</tr>
<tr>
<td>5.4</td>
<td>Mean Tardiness per Order as Affected by Flow Allowance, Shop 1, Order Type 1, U = .80</td>
<td>130</td>
</tr>
<tr>
<td>5.5</td>
<td>Mean Tardiness per Order for COVERT, Shop 1, Order Type 1</td>
<td>131</td>
</tr>
<tr>
<td>5.6</td>
<td>Mean Tardiness per Order With and Without Hold-off Heuristics, Shop 1, Order Type 1</td>
<td>132</td>
</tr>
<tr>
<td>5.7</td>
<td>Mean Tardiness per Order, Order Type 1</td>
<td>136</td>
</tr>
<tr>
<td>5.8</td>
<td>Mean Tardiness per Order, General Shop, Order Type 1, U = .8</td>
<td>137</td>
</tr>
<tr>
<td>5.9</td>
<td>Mean Tardiness per Order, Shop 1, Order Type 2, U = .84</td>
<td>139</td>
</tr>
<tr>
<td>5.10</td>
<td>Mean Wait and Staging Times, Shop 1, Order Type 2, U = .8</td>
<td>139</td>
</tr>
<tr>
<td>5.11</td>
<td>Mean Tardiness per Order, Shop 2, Order Type 2, U = .83</td>
<td>141</td>
</tr>
<tr>
<td>5.12</td>
<td>Mean Tardiness per Order, Shop 1, Order Type 3, U = .80</td>
<td>142</td>
</tr>
<tr>
<td>5.13</td>
<td>Summary of Hypotheses and Conclusions</td>
<td>143</td>
</tr>
<tr>
<td>5.14</td>
<td>Retrospective Experimental Design</td>
<td>144</td>
</tr>
<tr>
<td>D.1</td>
<td>Experimental Results for Single Component Orders, Single Channel Machine Groups</td>
<td>177</td>
</tr>
<tr>
<td>D.2</td>
<td>Experimental Results for Single Component Orders, Multiple Channel Machine Groups and the &quot;General Shop&quot;</td>
<td>178</td>
</tr>
<tr>
<td>D.3</td>
<td>Experimental Results for Multiple Component Orders</td>
<td>179</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Illustration Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Flow Time Mean and Standard Deviation Relationships for Selected Sequencing Rules.</td>
<td>61</td>
</tr>
<tr>
<td>3.2 Marginal and Conditional Flow Distributions</td>
<td>68</td>
</tr>
<tr>
<td>3.3 Conceptual Lateness Distribution</td>
<td>69</td>
</tr>
<tr>
<td>3.4 Linear Approximation to Probability of Tardiness</td>
<td>76</td>
</tr>
<tr>
<td>3.5 Example Routing for a Multiple Component Order</td>
<td>88</td>
</tr>
<tr>
<td>3.6 Typical Serial-Parallel Routing</td>
<td>92</td>
</tr>
<tr>
<td>4.1 Schematic Representation of the Experimental Plan</td>
<td>104</td>
</tr>
<tr>
<td>4.2 Gross Flow Chart for the Simulation Program</td>
<td>110</td>
</tr>
<tr>
<td>5.1 Lateness Distribution for Six Rules, Shop 1, Order Type 1</td>
<td>127</td>
</tr>
<tr>
<td>5.2 Mean Tardiness per Order for 24 Successive Blocks of 128 Successively Released Orders, Shop 1, Order Type 1</td>
<td>134</td>
</tr>
<tr>
<td>B.1 Flow Chart for GERES.</td>
<td>158</td>
</tr>
<tr>
<td>B.2 Flow Chart for COVERT</td>
<td>162</td>
</tr>
<tr>
<td>B.3 Flow Chart for CTPLS</td>
<td>163</td>
</tr>
<tr>
<td>B.4 Flow Chart for WORKS, Single Component Orders, Single or Multiple Channel Machine Groups</td>
<td>164</td>
</tr>
<tr>
<td>B.5 Flow Chart for CTPLS and WORKS, Multiple Component Orders</td>
<td>169</td>
</tr>
<tr>
<td>C.1 Flow Chart for Critical Path Update Upon Arrival in Queue</td>
<td>172</td>
</tr>
<tr>
<td>C.2 Flow Chart for Critical Path Update Upon Incurrence of Delay</td>
<td>173</td>
</tr>
<tr>
<td>C.3 Flow Chart for REDOCP (for Orders Which Are Early or Late)</td>
<td>174</td>
</tr>
<tr>
<td>C.4 Flow Chart for REDOCR (for Orders Which Are Tardy)</td>
<td>175</td>
</tr>
</tbody>
</table>
Chapter I
INTRODUCTION

Preamble

The purpose of this study is to find a better sequencing system for job shops. Further, it is hoped to find a system adaptable to some of the more common complexities of real job shops such as "multiple channel" service centers and multiple component orders (i.e., orders involving assembly operations). The plan is to develop and test a rule for a simple model and then to add specialized rules or "heuristics"\(^1\) to cope with the features of more complicated models. "Sequencing" is used here to denote the process of determining which of several eligible jobs should be performed next at a free work station. It refers to the ultimate exercise of choice regardless of the basis of the decision and therefore includes "scheduling", preplanning the sequence, as well as "dispatching", making the choice at the last moment when the machine becomes free. This work is thus directed towards finding better solutions for particular problems in sequencing by building on the foundation laid by previous studies. This pattern is succinctly described by Richard Bellman:

\[^1\]Heuristics are rules or procedures which lead, on the average, to improved solutions. For more profound definitions and discussion, see Marvin Minsky, "Steps Towards Artificial Intelligence", Proceedings of the IRE, (IL, January, 1961), 8-30.
"In entering new fields, such as turbulence, or adaptive control, or the many body problem, or scheduling, it is essential to solve as many particular problems as possible in the hope that patterns and solution and structure will emerge. Following initial clues, we solve further problems and so, hopefully and at least historically, converge to a theory."^2

It follows that no Newtonian breakthrough is found here.

The basis for the rule developed here is the so-called "c/t rule" which was shown by Wayne Smith and Robert McNaughton to minimize the total delay cost of n jobs to be processed on a single machine.\footnote{Wayne E. Smith, "Various Optimizers for Single Stage Production", Naval Research Logistics Quarterly, (III, March 1956), 59-66, and Robert McNaughton, "Scheduling with Deadlines and Loss Functions", Management Science, (VI, October 1959), 1-12.} The rule is beautifully simple: it calls for selecting the jobs in sequence by decreasing \( \frac{c_i}{t_i} \) ratio, where \( c_i \) is the delay cost rate for the \( i^{th} \) job and \( t_i \) is its processing time. The rub in applying this in a shop comes from the simplicity of the model from which the rule is derived. How one goes about assessing delay costs, considering other arrivals,\footnote{The c/t rule is also optimal for queues with exponential arrivals and arbitrary service times, it is shown by David Cox and Walter Smith in Queues, (London: Methuen, 1961), 83-86. But they do not consider the case of known arrivals which is implied above.} other "channels" in the same service center, subsequent tasks on the jobs at other machines, and the like, is not clear. Therein hangs this tale.

The experimental medium is digital simulation. This surely requires no defense; simulation is now a standard tool for system studies. But, the requirement for simulation not withstanding, the use is dangerous. An

\footnote{"Bigger Computers and Better Mathematicians", Datamation, (X, June 1964), 45.}
experiment in which the experimenter controls all variables, exogenous as well as endogenous, all interactions, all measurements—in short, everything—fairly cries for skepticism. For this reason, some pain is taken hereafter to describe the model, the program, the experimental plan, and procedure "for the reader's evaluation . . . in considerable detail".  

Another danger associated with simulation studies is the temptation to test procedures or disciplines purely on the basis of their superficial plausibility. The viewpoint adopted here is that mere plausibility is not enough; either a discipline has proven properties that make it desirable or can be shown (i.e., proven) to work well in particular situations. In nearly every case where a procedure is proposed here a theory of its behavior will be provided.

Among other things that have motivated this study is the increasing interest in and application of "on-line, real-time" control systems for manufacturing. An important feature of these systems is that they permit bringing currently accurate "global", i.e., central file, information to bear in making "local" decisions. The utility of this feature for airlines making reservations for a single flight from many remote offices is obvious;

---

5 Quotation from Richard W. Conway, "Some Tactical Problems in Digital Simulation", Management Science, (X, October, 1963), 61. This is an articulate expression of the type of skepticism meant.

6 These are systems in which the activity to be controlled is connected by an information channel to the controlling agency ("on-line"), a computer, say, and in which the controller's response (direction) to a stimulus, i.e., status report, from the activity is forthcoming within a time small enough such that the status will not have changed materially.

they can avoid "overbooking" the flight. In a shop, access to a current central file would enable (among other things) consideration of the status of all jobs and all machines in making the choice of next job for one machine. The utility of this is not established.\footnote{An evaluation (by simulation) of a related situation, however, is reported in D. F. Boyd and H. S. Krasnow, "Economic Evaluation of Management Information Systems", \textit{IBM Systems Journal}, (II, March, 1963), 2-23.} Since the system proposed here does require such access, a by-product of the study is an assessment of the value of "global" information \textit{vis a vis} "local" information.

On-Line, Real-Time Systems and Incomplete Information

The implications of real-time system technology for sequencing system research is clear. No longer need research be directed only to rules which can be applied directly at the work station. It becomes technically feasible to implement highly complex rules using any information about current shop or job status and attributes. The horizon is thus limited only by the ingenuity of the researcher and, of course, by questions of economic feasibility.

But there is another facet of this. Considerable effort has been devoted in the past to the process of developing "schedules", that is, planned sequences of jobs at machines, promulgated considerably prior to the execution of the plan. It is asserted that the scheduling process is an attempt to solve the global problems of sequencing, that is, an attempt
to consider information that exists beyond the limits of the individual work station. But scheduling is at best an "incomplete information" process. By this is meant that information on events which occur between formulation of the plan and the time of execution is perforce missing. Suppose an "optimal" schedule is issued. Then, after a time, a machine breaks down, a forging cracks, a task takes longer than expected. The optimal schedule is likely to be no longer feasible. Either a new schedule has to be generated or a set of adjustments made to the original on the spot, i.e., in real time.

This point is raised because it is germane to the question of what are "better" sequencing systems. After all, there have been proposed integer linear programming formulations for the scheduling problem. Schedules generated by a program are optimal, albeit by limited criteria and subject to severe computational limitations. But they are optimal on the basis of incomplete information. The scheme proposed here is less sensitive to problems arising from the occurrence of unexpected events because decisions are delayed until the last moment and, as a consequence, are based on complete information on current status. Decisions are subject to errors of prediction of future events, of course, but all of the interim variations associated with a preplanned sequence are avoided.

In order to compare a real-time sequencing system with a scheduling system, it is concluded, one must consider the real-time application of the latter.

---

What is Known: A Thumbnail Assessment

Job shop sequencing has been the subject of widespread and intensive study during the past decade, there having been published over two hundred articles, reports, and theses related to this subject during the period. Some reasons for the interest are these: first, the job shop is the manufacturing modus operandi for a large sector of the economy, and sequencing procedures have been shown to have significant economic effects on shop performance; second, the job shop model or its abstraction, the queueing network, is ubiquitous, it applies as well to the problem of multiprogramming in computers, to office management congestion problems, or to the scheduling of research and development projects with limited manpower or facilities, as it does to the stereotypic machine shop filled with ringlets of metal shavings and the odor of cutting oil; and third, the analytical problem associated with queueing networks is so rich as to be attractive like Everest, because "it is there".

Despite this notably large research effort in sequencing, most research, even based on simulations, has been limited to studies in depth of very simple situations. In particular, the studies have centered on shops with single channel service centers, orders with one component (no assembly operations) and, usually, very simple sequencing rules.

There have been six major simulation studies of sequencing: by Alan Rowe,\textsuperscript{10} "Towards a Theory of Scheduling", \textit{Industrial Engineering}, (XI, March, 1960), 125-136.

\textsuperscript{10}
Richard Conway, Bruce Johnson and William Maxwell, 11 Conway and Maxwell, 12 William Gere, 13 Conway again, 14 and by Yves R. Nanot. 15 In every case, simple orders and simple shops were studied; in most cases simple rules were studied. Rowe permitted multiple channel service centers in his model but did not consider them as a factor in his design.

The major findings of these studies can be summarized as follows:

1. There are significant differences (practical and statistical) in shop performance attributable to the sequencing rules used (all).

2. The shortest imminent operation rule, i.e., select next that job requiring the least processing time, results in lower mean waiting times, shorter mean completion times for orders, and higher capacity utilization than other rules tested (most thoroughly investigated by Conway and Maxwell but confirmed by others).

3. When utilization is allowed to vary (say by maintaining a constant number of orders in the shop), the shortest imminent operation rule produces a very high variance of order completion times (Conway and Maxwell).

4. There is no significant difference as to mean wait times and order completion times among rules ignoring processing times (Conway, Johnson, and Maxwell).

5. Rules based on order lateness ("due-date oriented rules") tend to reduce the variance of the order completion time distribution, the mean remaining relatively unchanged (Rowe, Conway, Johnson, and Maxwell, Nanot).


6. Rules based rationally on order value reduce mean work-in-process inventory (Rowe; Conway, Johnson, and Maxwell). Assuming orders can be shipped when completed, the shortest imminent operation rule provides very low work-in-process inventory (Conway).

7. A good balance between conflicting inventory level and order tardiness goals can be achieved by starting high value orders later and sequencing with a due date oriented rule (Rowe).

8. The use of heuristics to modify simple rules based on specific features of the service center, queue, or task, leads to improved performance (Gere).

9. Due-date oriented rules which are "dynamic", i.e., change with time, tend to "make-up" initial earliness or lateness of orders, and thereby to reduce the variance of completion times (Nanot, based on analysis and exploratory work of Jackson, see below).

10. Using a fixed utilization level model, the shortest imminent operation rule performs surprisingly well as to percent of tardy orders, but not as well as Rowe's "slack per remaining operation" (Rowe and Conway).

The results seem remarkably insensitive to environmental factors such as "load", i.e., steady state utilization, shop configuration, shop size, variability of order routings, and errors of estimates in processing time.

In addition to these simulation studies, there has been considerable analytical research (usually involving very simple models). Aside from the afore-mentioned studies of Smith and McNaughton, of direct interest is the work of Michael Rothkopf who extended the McNaughton approach for a variety of complications such as arbitrary delay cost functions, multiple channels, and interruptions of service, and also the many studies of Jackson on "job

---

shop-like" queueing networks and dynamic priorities. Jackson's work supplies what comes closest to being a phenomenological theory of job shops.

The body of knowledge accumulated to date is impressive; a solid foundation has been laid for further work. A great deal has been learned about the bits and pieces from which more sophisticated approaches may be fabricated.

It is contended, however, that very little of direct operational value has been produced so far. This contention consists of two parts, namely: that research has centered on models too simple to be representative of the majority of real-life job shop problems and that the nature of the real-life complexities changes the nature of the problem sufficiently to call for sequencing rules which take into account the complexities themselves. For example, the assumption of single channel service centers is convenient, yet one suspects that most shops have more than one machine of a type. And, it has not been established that rules which are "good" for single channel centers are also good for the multiple channel case.

A more important example may be found in the assumptions on the orders. Single component orders are basic and simple to deal with; their routings are serial, only one operation at a time can be performed on the order. Operations interact only in competing for limited capacity. It is asserted, however, that multiple component orders with assembly operations are more common. Indeed, in two of the simulation studies noted above, the authors cite as their

---

"typical" job shop orders, ones involving multiple components. It seems intuitively clear that rules which take into account multiple component information such as relative lateness of components (something similar to "critical path" information) are likely to sequence better than rules which ignore it. It is so conjectured.

Holt's Proposal

It would certainly be remiss not to cite Holt's proposal for "minimizing the cost of queues". In a nutshell, he suggested a two stage sequencing system: in the first stage a "global analysis", perhaps (he thought) reducing to quadratic programming, is used to derive optimal release and starting times for operations; in the second, in real time, a type of c/t rule is used to select tasks for machines. He unfortunately did not submit his proposal to testing. But the basic form of the rule proposed here is at least a philosophical first cousin of Holt's.

Delineation of Problem and Model

In order to test the rules which are proposed, a rather simple model will be used both to provide for experimental feasibility and compatibility

\(^{18}\) Rowe, op. cit., 126, and Gere, op. cit., 9.

\(^{19}\) "Priority Rules for Minimizing the Cost of Queues in Machine Scheduling", Chapter 6 of Muth and Thompson, op. cit., 83-95.

\(^{20}\) A similar but simpler rule was suggested by me in an internal memorandum at the Manufacturing Controls Department, Westinghouse Electric Corporation, July 11, 1960.
with previous research. For instance, the shop will be limited in size to eight service centers which are assumed to be manned continuously. Machines do not break down. Routings (sequence of machine assignments) and assignments themselves are inviolable. Processing times for tasks are known \textit{a priori} with certainty and include intermachine transit times and setup times. Orders arrive randomly (but with a stationary arrival rate following Nanot rather than Conway and Maxwell). Processing times for given machines are exponentially distributed.

The routings of orders are obtained from a "transition matrix", \( P \), containing as elements \( p_{ij} \) the probability of a task at center \( j \) following a task at center \( i \). Conway and Maxwell define a \textit{pure job shop} as being characterized by a matrix with equal elements off the main diagonal (the main diagonal being zero corresponding to no "repeat" tasks). A \textit{pure flow shop} is defined as one characterized by all zeros and ones such that "each machine has a fixed predecessor and successor". The routing used here will be very nearly that of a pure job shop.

The messy question of performance criteria is handled in the following manner. The mean level of utilization is held fixed by fixing the mean

\footnotesize

\begin{itemize}
  \item For the more complicated models, the heavy machine shop at the Steam Turbine Division of Westinghouse Electric Corporation supplies the inspiration. There is no quantitative relationship between this very large shop and the model, but it is offered that a qualitative expression of the sequencing problem there has been achieved.
  \item \textit{Ibid.}
\end{itemize}

\normalsize
arrival rate and mean work content of orders, and an assumption is made that orders completed early, i.e., before their "due dates", must be held until delivery. Also orders are assumed to be of equal value. Release times are not a function therefore of order value. Under these conditions, the only variable cost is the "tardiness penalty". For comparing performances in experiments all orders are assumed to incur linear tardiness penalties at the same rate. There is, therefore, a single performance measure for comparing sequencing rules, namely, mean order tardiness.

Because previous studies have generally indicated that results are insensitive over a broad range of environmental conditions, a broadly factorial experimental design has been avoided. However, in addition to the sequencing rules themselves, the shop configuration (as to channels), the order configuration (as to number of components), the mean (steady state) level of utilization, and the "tightness" of the due dates, are considered as factors.

Some Definitions

Pursuant to discussion of job shop problems at a more substantive level, the following definitions are offered. A shop consists of a set of machines (manpower is certainly there, too, but is not considered directly). Subsets of identical machines are called machine groups. The term work station and service center are used synonymously. Multiple channels, borrowed from the vernacular of queueing, refers to multiple machines of a machine group.
A shop produces orders as output, from raw materials, the input. An order has associated with it a routing which is a set of ordered tasks or operations to be performed at machine groups. The ordering restrictions or precedence relationships within the routing require the completion of one or more tasks before another may be started. (Note that the term "job" is avoided. It is often applied to mean both the task and the order; its use is reluctantly abandoned hereafter.)

An order consists physically of one or more components, e.g., pieces of metal. If an order consists of only one component and the sequence of operations upon it is uniquely specified, this is called a serial order. Except when noted, all single component orders are taken to be serial orders and have serial routings. If, on the other hand, an order consists of two or more components which must ultimately be assembled before the order is completed, (implying that tasks on different components of the order may be performed simultaneously), and the sequence of operations on each component is uniquely specified, this is called a serial-parallel order, or a multiple component order. The routing is said to be serial-parallel.

By environmental assumption, each order is assigned a due date, this being that point in time after which completion of an order is considered tardy and before which completion is considered early. Note that lateness is hereafter reserved for the status of an order still in process vis a vis some plan or schedule. The time of arrival of the raw material for a component of an order in the shop is called its release time or release date.

Now, since tasks take finite processing times and the number of machines is limited, there exist queues of tasks awaiting processing. For multiple
component orders there exists an additional delaying state which occurs when components are ineligible for processing because other components have not completed their tasks which precede the assembly. The state of the waiting pieces is called staging, the delay, staging time. The term waiting time unless otherwise noted will be used generically. The total elapsed time from (earliest) release of an order to its completion is its flow time.

Given a queue at a machine group, when a machine completes a task, one of the several eligible tasks must ordinarily be selected. The selection procedure is the sequencing rule. The process is called sequencing or dispatching. A plan which associates with each operation of an order's routing a date at which that operation is (theoretically) to start is called a schedule, the dates, start dates.

Outline of Material Which Follows

Chapter II is devoted to establishing a context for sequencing problems in the hierarchy of job shop production planning and control problems, and to classifying the voluminous research in this field by problem type and approach. Research of direct relevance to this study is reviewed within this classification.

In Chapter III, a rationale for sequencing is developed, building first on previous results and then synthesizing rules to extend to problems not

24 The possibility of purposeful idle time, "holding off", is to be examined.
previously treated. A digression is taken to present a formal description of multiple component orders.

Chapter IV is devoted to a statement of the detailed hypotheses on the behavior of the proposed rules, a description of the model as implemented in the simulations, and a full statement of the experimental design and procedure.

The results of the experiments and a direct interpretation of the results are presented in Chapter V; and in Chapter VI, broader conclusions are drawn and suggestions for further study are outlined.
The Job Shop Environment

While the denotation of a job shop by variability of routing serves the purpose of differentiating among manufacturing situations, it hardly conveys the full flavor. There are several connotations to the term job shop which are important here because they underlie certain problem assumptions to be made and explain some aspects of the performance criteria.

There are connotations as to the supplier-customer relationship, for instance. In a job shop, it is usual for the customer to place an order for a specified product which is then built to these specifications and completed, if all goes well, by a date, the "due date", negotiated with the supplier. This implies that the manufacturer has little or no finished goods inventory and work on an order does not commence until the order is received. Of course, almost all manufacturers carry certain basic common items ("nuts and bolts") in inventory, and the recent trend toward "modular" products which permit assembly of "tailored" finished products from stocked major components both serve to modify this situation. Nonetheless, in what is called here a job shop, it is assumed that the "made-to-order" situation predominates. Furthermore, if there are questions of lot size, the manufacturer has little or no option. This follows from the uniqueness of the finished product.

A further connotation of this "made-to-order" mode of operation is that the shop is capable of producing a wide repertoire of products in order to cater to the whims of diverse customers. And this implies that the equipment
in the shop will likely be general-purpose rather than special-purpose. That is, one would expect to find engine lathes, radial drill presses, and planers rather than specialized engine block drillers, for example. Also, the diversity of product tends to result in a layout or arrangement of the plant according to the type of task performed rather than the particular product made, in order to concentrate related special skills. Finally, the diversity of products, functional layout, and general purpose equipment combine to imply variable routing of products through the factory, which completes the circle. ¹

The Planning, Scheduling, and Sequencing Hierarchy

As in most materials flow problems, there exists in the job shop a number of levels at which the problem may be perceived. And, in common with these problems, the constraints at lower levels are controllable variables at higher levels. In this case, work station capacity is taken as fixed, as a constraint, but at a higher level, the capacity planning problem itself could be studied.

At the top of the hierarchy is what might be called long-range planning. At this level, virtually the entire environment is variable. Major changes in the composition of the work force may be contemplated, additions of new equipment and changes in factory layout are all possible.

At the lowest level, the real-time decision level, essentially nothing is variable except the choice of task for a given work station or work

¹Rowe, op. cit., and Gere, op. cit., both have lengthy and informative discussions of the job shop environment. See also "Improved Job Shop Management Through Data Processing", (New York: IBM Corporation, 1960).
station for a given task. Even so, there may be several degrees of freedom in these decisions. For instance, if the specified sequence of operations to be performed on an order is not fixed, a decision is required as to which operation on the order to perform next (or which of several queues to join). This will be called the alternate routing problem. Or, if certain tasks can be performed by more than one machine group, another type of decision must be made, namely, which of several machine groups to perform the specified task. This will be called machine substitution, the freedom being derived from machine capabilities as opposed to options within the order routing. In many cases the possibility of "bumping", terminating one task before it is completed in order to process a more important task, or "splitting", processing only a portion of a lot, may be considered. In all these cases, the choice among alternatives is ordinarily made immediately before execution and is based on current status of queues, expected waiting times, order lateness, status of machines, and the like. ²

Even if none of these situations obtain, there still remains the fundamental real-time decision, that of dispatching, of selecting one from several eligible tasks to be performed next when a machine becomes free. This is not a simple decision. For a single machine, with n eligible tasks to choose from, there are n! possible alternative sequences to be considered. (Of course, the decision maker can only select one of n+1 tasks, including none at all, but it is assumed that he must consider the implications of his

²None of these problems are studied here. However, two master's theses in process by Frank Russo and Paul Clermont deal explicitly with alternate routing and machine substitution. Their work is based on the model and general experimental plan set forth here, and is described in Chapter VI.
initial choice on all other tasks which are known to be delayed by the one he selects.)

Between the planning level, where nearly everything is variable, and the real-time level where only the next task performance is variable, there are a number of important decisions. The capacity of the shop must be established (within gross constraints established at the planning level). Decisions as to furlough and recall, reassignment of workers, timing of machine maintenance, and in the shorter run, overtime (and undertime) must be made. Also decisions affecting load are made. These include the bidding decisions of the firm, and subcontracting of tasks from overloaded work stations. Bidding certainly also includes the problem of due date establishment. It is thus that the specific plan of action ultimately to be executed via numerous real-time decisions must be prepared. The process of capacity adjustment is called here structuring, that of load planning, including due date establishment, is denoted as master scheduling. To digress, it is remarkable how little research has been brought to bear on the problems of long-range planning, structuring, and master scheduling. It should be granted that with some alterations, the "linear decision rule" can be applied for seasonal planning of employment but almost no explicit consideration of higher level job shop problems has occurred.

It is not intended to study these problems directly here. But an attempt

---

3 C. Holt, F. Modigliani, and H. Simon, "A Linear Decision Rule for Production and Employment Scheduling", Management Science, (II, October, 1955). The rule presumes finished goods inventory which does not apply to job shops generally. The buffering effect has to operate through use of "backlog" or work-in-process inventory.
will be made to trace the implications of the lower level decisions under study on higher level problems.

As has been noted, decisions made at these higher levels result in the constraints under which the low-level decisions must be made. But the interaction is clear: the low-level decisions establish the basic operating character of the shop, that is, how it tends to behave within the various constraints. One might reconsider this hierarchy of problems in a linear programming framework. The low-level decisions result in a solution to the constrained problem. The higher level problems then would be analogous to "parametric programming" problems. Insofar as this analogy holds, it would appear expedient to seek solutions to the constrained problem before generalizing to study the constraints themselves.

**Economic Criteria Problems**

In no place does the interaction of decisions cause more difficulty than in discussion of economic criteria. At the "fully constrained" level, assume due dates and release times are fixed, manpower constant, and the arrival rate of orders fixed. In this situation there are only two aspects of performance which are variable with the remaining decisions: order tardiness and inventory level costs. Moreover, the latter apply only in the rather rare case (in my experience) in which substantially early shipments of completed orders are permitted without rancor.

But, consider the case with early shipments permitted. The cost generating activity is delay; any time an order is delayed an "avoidable" inventory cost is incurred. Perhaps the most legitimate framework is to consider a delay a postponement of capital accession. In this sense, the cost of delay
is the full price of the order forgone for the length of the delay (but discounted by the time until earliest possible delivery, for purists). This applies regardless of the tardiness status of the order. Additional costs accrue after tardiness but their assessment is impossible in the absence of institutional assumptions. In some instances, "deadline losses" apply; there are fixed costs incurred upon entering the tardy state. In some cases, contracts provide for explicit penalties, most frequently at a constant rate. With rare exception there are goodwill losses, both with the customer for a promise not kept and as an effect on general reputation. Observations of management behavior in several shops lead me to believe that these intangible costs are often very high.

In the case (asserted here to be a more valid general assumption) in which early shipments are not permitted, all variable costs are functions of order tardiness. With arrival rate and work force fixed, the utilization level is determined. A system which attempts to reduce idle time has utility only insofar as it reduces tardiness by better exploitation of available capacity. With release time and due dates fixed, the inventory level is fixed, except for tardy orders. However, tardiness costs in this case should include forgone capital charges as well as direct losses.

Two models, oriented towards low-level studies, have varied from the two above. One explicitly considers release times as part of the "scheduling" system. The other is based on a "constant backlog" assumption, namely that the arrival rate can be varied such that a constant number of orders are in process.

---

4 Rowe, op. cit.
5 Conway and Maxwell, op. cit.
These are considered in turn.

If release times are set as part of the system (the only interesting case is where early shipments are prohibited or penalized), then inventory levels can vary with the choice of release times. In fact, a direct trade-off exists in this case between tardiness risks and inventory level. The fundamental idea is to avoid earliness without increasing tardiness too greatly. The saving comes from postponing investment in materials (and labor, assuming the labor force is variable) until as late as possible. Obviously, it makes sense to adopt a strategy in this case of releasing more valuable orders later than low value orders: the higher the value, the faster the turnover.

The Conway-Maxwell model also involves a trade-off situation. A system which produces low mean flow times will, in their model, have higher utilization. The faster orders can be turned out, the faster they will be brought in. Since a constant number of orders are held in the shop, inventories are roughly the same for all systems. But the low mean flow time system will produce highest "turnover" because it will produce more orders per unit time. The trade-off occurs with tardiness, it seems. With such a model, the variability of flow time (or more precisely the "right hand tail") of the flow time distribution appears to be large when the mean is small.

Approaches to Research

Even the lower level problems of job shops are rich enough to have evoked a variety of approaches to solution of myriad versions of the problem. An attempt is made here to classify these according to three basic criteria.
First, the fundamental approaches to sequencing of tasks on machines appear to fall in one of two categories. An attempt is made either to derive a schedule "all-at-once", and then to enforce this schedule as closely as possible (it is presumed) in assigning tasks to machines in real time; or alternatively, to derive a sequencing rule, which may be applied directly in real time. The Gantt Chart is the prototype of the "all-at-once" approach. With this device one lays out time-proportioned strips representing tasks on time scales representing machines. One carries this process well into the future, often for the length of time needed to complete the tasks for all orders currently in backlog. Since one uses the estimated times for each task, observes all precedence constraints (sometimes with difficulty) and work station capacity constraints, one obtains from the process of "solving" the Gantt Chart a theoretically feasible schedule which, depending on the skills of the user, may or may not be effective in terms of meeting due dates, or achieving such other goals as may be appropriate. When a machine actually becomes free, the chart is consulted to see which task should be undertaken next.

This mode of operation applies as well to other "all-at-once" schema such as schedules generated by linear programming or Monte Carlo generation, or heuristic "layout" procedures. That is, a schedule is generated then, in real time, sequencing is accomplished by selecting the task with the earliest scheduled time for processing.

---

That which distinguishes the "all-at-once" approaches is the fact that the sequence of tasks at a particular work station need not be explicitly considered. For this reason, these approaches will be called implicit sequencing.

The other approach, explicit sequencing, includes not only application of simple priority rules but also heuristic rules based on situational patterns at the work station, the rest of the shop, the particular orders involved, or other orders. The salient feature is the presence of a decision process yielding a unique choice of the next task to be performed. One often finds, both in industry and in the literature, use of what is called a schedule using non-trivial sequencing rules (i.e., something other than earliest scheduled date). However, this schedule ordinarily serves the purpose of providing only approximate measurement of order lateness and is typically based on simple "backdating" from the order due date using estimated processing times and estimates of wait times. In this sense it is not a feasible schedule but rather a convenient repository of information elements of relevance to the sequencing process.\(^7\)

Emery has proposed a hybrid system in which simulation using an explicit sequencing system is used to generate a feasible schedule,\(^8\) the idea being that the feasible schedule is used for real-time dispatching by earliest scheduled

---

\(^7\) The elements being remaining processing time, perhaps remaining allowance for waiting time, and due date. These elements could be supplied directly to the rule, hence the equivalence of "minimum slack" and "earliest scheduled start date" rules when the latter is due date less remaining processing time.

\(^8\) Ibid., 85 et. seq.
date. This incidentally supplies a second means of implementing very complicated rules such as some of those to be proposed. In this mode of operation, the sequencing system is subject to the normal "incomplete information" problems, however.

The research in job shop sequencing can be further subdivided by the scope of the problem analyzed. A favorite subject of study has been the single work station. Searching for generality, some researchers have looked at two or three centers or multiple channels at a single work station. Although one cannot usually generalize about system behavior on the basis of one or two component analysis, understanding of the system usually implies some understanding of component behavior. These component studies can thus be considered as productive of knowledge which is necessary but not sufficient for understanding systems. Any study in which the results cannot be directly generalized to larger systems shall be denoted as microanalysis. Where an attempt is made to obtain generally applicable conclusions, the study will be classified as macroanalysis.

One other classification seems appropriate, namely that relating to the time dependence of the system (including one component systems) studies. By a static system will be meant one in which no new arrivals of orders, tasks, or for that matter, work stations are permitted. In macroanalytic studies, this implies the orders to be scheduled or the tasks to be sequenced are all present and known at the beginning of the period. In static microanalysis, new arrivals into the queue are not permitted. Dynamic systems include garden-variety queueing studies as well as system simulation studies permitting stochastic order arrivals.
Using these three criteria for classification, the following fundamental model types can be identified:

1. **Static microanalytic studies**: the "n-job" problems wherein the sequence of tasks with no new arrivals in queue is considered for a single channel, multiple channels, or two or three stations; explicit or implicit (e.g., dynamic programming) orientations are both found.

2. **Dynamic microanalytic studies**: single or multiple channel queueing studies; only explicit sequencing, "queue discipline", is studied.

3. **Static macroanalysis**: queue networks with all orders released at initiation; these are predominantly implicit sequencing studies.

4. **Dynamic macroanalysis**: queue networks with stochastic order arrivals; attempts to generalize about transient or steady state behavior; explicit sequencing using simulation is the usual mode.

The literature to be discussed will be categorized in this way.

**Common Assumptions**

Sisson has listed for a general model the assumptions which are typical for studies in this area. They are (substituting terms defined here in case of conflict and supplying free translations):

1. No machines may process more than one task at a time.

2. Each task, once started, must be performed to completion (no "bumping", all machines manned continuously and no breakdowns).

3. An order is an entity; that is, even though the order represents a lot of individual parts, no order may be processed by more than one machine at a time (no "splitting" and no parallel routings).

4. A known, finite time is required to perform each task and each task must be completed before any task which it must precede can begin (no "lap phasing").

---

5. The time intervals for processing are independent of the sequence in which tasks are performed.

6. Each order must be processed by a designated sequence of machines, this sequence being also called "the technological ordering" or "the routing" (no alternate routing, no machine substitution).

7. There is only one of each type of machine (no multiple channels or machine groups).

8. A task is performed as soon as possible (no "hold-off").

9. All orders are known and are ready to start processing before the period under consideration begins (i.e., static analysis).

10. The time to transport an order between machines is negligible (move time ignored).

11. In-process inventory is allowable.

There are important published exceptions only to assumptions 8 and 9, although relaxation of assumption 5 for the single station static backlog can result in the familiar "traveling salesman" problem.

Criteria have always caused problems. Tardiness, even in simplified forms, does not lend itself to simple mathematical expression [it is, in fact, \( \max(o,f-a) \) where \( a \) is "allowable flow time" (due date less release time), and \( f \) is actual flow time]. When the model is more general such that inventory or utilization considerations apply, the problems become even more intractable.

Faced with this problem, researchers have usually selected one of these four strategies:

1. Conduct a strictly phenomenological study. Measure all of the relevant aspects of performance for each scheme, but avoid value judgements among the schemes.

2. Use an approximate performance measure such as the "minimum makespan", total time to complete a given set of orders, which is reasonably consistent with more precise criteria (see below).

3. Concentrate on one or two of the three aspects of performance such as utilization or tardiness.
4. Assume an institutional setting and provide normative conclusions for the setting.

Of these, the second has been that strategy chosen most frequently for the more abstract research. Some of the criteria have been:

1. minimum make span (as defined above)
2. minimum sum of completion times
3. minimum maximum tardiness
4. minimum mean wait time
5. minimum percent of tardy orders.

As these can be quite different, one must take care in comparing strategies addressed to different objectives.

The Literature

In that three major reviews of the literature in this field have been published, only that research which bears directly on the work in this thesis is discussed.\textsuperscript{10,11,12}

Static Microanalysis: The N-job Problem

Among the earliest attacks on the "n-job problem: was Jackson's.\textsuperscript{13} He


\textsuperscript{11}Gerald L. Thompson, "Recent Developments in the Job Shop Scheduling Problem", \textit{Naval Research Logistics Quarterly}, (VII, December, 1960), 585-589.

\textsuperscript{12}"Sequencing Theory", \textit{op. cit.}

proved that by performing tasks in sequence of increasing due date, the maximum tardiness was minimized. Smith used Jackson's result in a two-phase approach for limited cases which results in minimum make-span, given that tardiness is eliminated altogether.\textsuperscript{14} Smith's procedure is as follows:

1. Establish that tardiness can be completely eliminated by Jackson's rule.

2. Select for the last unfilled position in the sequence that task which has, of all those tasks which will not be made tardy by being processed last, the largest processing time.

3. Repeat step 2 until all \( n \) tasks are sequenced.

McNaughton looked at the same problem with some different performance criteria.\textsuperscript{15} He, like Smith, obtained the so-called "c/t rule" for linear delay costs. This may be stated as follows. Let \( c_i \) be the (given) cost per unit time of delay of the \( i \)th task, and let \( t_i \) be its processing time, \( i = 1, 2, \ldots, n \). The total delay cost for the \( n \) tasks is minimized if the tasks are processed in sequence of decreasing \( c_i / t_i \) ratio. The rule applies only for single stations. Parenthetically, Conway and Maxwell state this result in a different form: namely, that the rule minimizes any linear function of completion times.\textsuperscript{16} Specifically (using their notation), if \( F_i \) is the completion time of the \( i \)th task, \( v_{ij} \) is the value of the \( i \)th task of the \( j \)th property, e.g., urgency, \( a_j \) is the weighting of the property, and the objective

\textsuperscript{14} Op. cit.

\textsuperscript{15} Op. cit.

is to minimize

\[ \sum_i \sum_j a_j v_{ij} F_i \]

then the tasks should be sequenced in order of decreasing

\[ \frac{\sum_j a_j v_{ij}}{t_i} \]

ratio.

This important theorem unfortunately has limited application in situations where tardiness is of importance in that it applies only when all n tasks are already tardy and tardiness penalties are linear.

McNaughton also looked at the problem of sequencing tasks with "deadlines", (due dates with only fixed penalties for tardiness). He established that an optimal schedule exists in this case, but did not discover an efficient procedure for obtaining it. (Several people have formulated this as a dynamic programming problem, see below).

A recent attack on the n-job problem in numerous versions was launched by Rothkopf. He found, among other results, a rule providing minimum delay costs for discounted delay cost rates of the form \( c_i(t) = c_i e^{-rt} \) for single stations and an approximate \( c/t \) rule for non-linear delay costs.

The work of Jackson, Smith, McNaughton and Rothkopf has resulted in sequencing rules which yield optimal or near optimal schedules for a variety of criteria. In addition, several researchers have formulated implicit sequencing approaches. Schild and Fredman have proposed an algorithm for the n-job case which provides good (but not necessarily optimal) performance by a

\[ ^{17} \text{Op. cit.} \]
minimum tardiness criterion. They also have obtained a dynamic programming formulation for arbitrary tardiness cost functions. Held and Karp achieved essentially the same formulation independently. Rothkopf has provided dynamic programming formulations for a variety of versions of the problem.

For all this impressive effort on this rather limited problem, it is notable that no rule has been developed which is computationally desirable and which gives optimal performance on order tardiness.

Multiple Stages and Multiple Channels

Very little research has been focused on sequencing with multiple channels. Rothkopf formulated a dynamic programming approach for cases when all channels are started empty as well as for the delayed channel availability case. He considered both the linear delay costs and absolute deadline cases, and also showed that the c/t rule is not in general optimal for this case.

More attention has been given the multi-stage problem, that is, the case in which orders are routed to two or more work stations in series. The general assumption is made that all orders have the same routing (Conway's


"pure flow shop"). The earliest attempt was Johnson's algorithm for two stages which he extended to three for restricted situations. His scheme is this:

1. Find the shortest processing time regardless of machine;
2. If the shortest time is on the first machine, place that order first in sequence;
3. If the shortest time is on the second machine, place that order last in the sequence;
4. Remove the order from the list to be schedules, and repeat.

Jackson extended Johnson's approach to the case in which the orders have different routings. Mitten looked at the two machine problems in which tasks on the order are separated by known arbitrary time lags, and found a delay minimizing rule. Johnson's rule also forms the basis of a heuristic approach for the general three machine cases.

Mitten's approach is operationally useful, he point out, in cases where two work stations dominate the scheduling problem; that is, where there are two "bottlenecks". The stations which come between the bottlenecks on the routing of orders have sufficient capacity that they do not

---

22 Selmer M. Johnson, "Optimal Two- and Three-Stage Production Schedules with Set-up Times Included", Chap. 2 of Muth and Thompson, op. cit., 13-20.


affect the sequence, it is assumed.

**Dynamic Microanalysis: Queueing Studies**

The next step up in complexity occurs when the assumption of no arrivals is relaxed. For single (or a few) stations this study falls within the purview of queueing theory. However, despite the voluminous material generated in this field of study, little of direct relevance to job shop problems had been done until Cobham introduced the notion of arbitrary priorities into the literature. He concerned himself with developing waiting time statistics for tasks grouped into priority classes for single and multiple channel queues with exponential interarrival and service times for the tasks.

Jackson has made extensive studies of queues with "dynamic priority disciplines", those in which priorities change with the passage of time. The family of "scheduling enforcing rules" includes several of these, e.g., earliest scheduled start date. He obtained bounds for orders in different urgency classes (corresponding to different initial latenesses) in an earliest scheduled date discipline. This study led to generalizations for queue networks for priority systems of the same type (for which see below).

Phipps extended Cobham's results for discrete priority classes to a continuum of classes, and in particular to the case in which priority is inversely related to processing time (i.e., "the shortest operation rule"). This rule provided reduced mean wait times (relative to first-come, first-

---


served) for Poisson arrivals and arbitrary service times he found. Cox and Smith extended this result. They state it as c/t rule: namely, select the task with minimum c/t ratio, mean delay costs will be minimized thereby.

Static Queue Networks

An important amount of effort has been directed towards the problem of scheduling a fixed set of orders through an arbitrarily large shop using various performance criteria and including "satisficing" as well as optimizing. The studies differ as to assumptions on shop configuration and order routing complexity, but the challenge is the same, lay out the "best" or a "good" schedule.

Of the most interest recently have been linear programming formulations based on integer constraints. Two different formulations were arrived at independently and essentially simultaneously by Bowman and Wagner; Manne added a new formulation, based on Bowman's, shortly after. All three seek to minimize the "make-span" and can accommodate due-date constraints as well as serial-parallel routings. However, the "machine-interference" constraints as stated apply only to single channel work stations. Wagner's

---


formulation was tested computationally by Story and him, and their experience resulted in some disappointing evidence regarding computational feasibility of the approach. As suggested by Muth and Thompson, "Manne's formulation, the most compact of the three, would apparently require [for three orders and four machines] 31 variables and 94 constraints....additional constraints are added during the course of its solution." So, it would appear that computational feasibility as well as the static nature of the model leave room for alternative approaches. An entirely different tack was taken by Giffler and Thompson. Their approach was to generate systematically all "active", feasible schedules, these being schedules in which no machine is held idle long enough for an idle task to be performed and feasible assignments are made as early as possible. This approach is computationally feasible, as tested by the authors and VanNess, only for small shops. For larger shops, they use a Monte Carlo technique for sampling active, feasible schedules to obtain a procedure which would tend to provide as "good" a schedule as sample size costs would permit. Their experience with random choice in competition with the "shortest imminent operation" rule disclosed that the latter gives schedules "slightly superior to the average shortest schedule [of the former]", over a large range of variability in processing times.


34 In "Introduction", Muth and Thompson, op. cit., xi.


A variation on the Monte Carlo theme was attempted by Fisher and Thompson.\footnote{H. Fisher and G. L. Thompson, "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules", Chap. 15 of Muth and Thompson, \textit{op. cit.}, 225-251.} Instead of selecting tasks randomly, they selected the rule for selecting the task randomly. Specifically, they chose between a shortest operation rule and longest remaining time rule on the basis of random numbers. They also included experiments in various types of reinforcement learning. They concluded that combinations of rules are better than exclusive application of separate rules and that learning is possible albeit not as important as the combination effect. Their goal was to minimize the make span.

Also in this study is the first application of a "hold off" rule, in which none of the waiting tasks is assigned so that a more critical task yet to arrive might be served.

Gere attacked the same static minimum make-span problem using rather complex heuristic sequencing rules (not in probabilistic combinations).\footnote{Op. cit.} He found that very simple sequencing rules augmented by his heuristics performed at least as well as, and in some cases better than, the best schedules generated by the Giffler and Thompson Monte Carlo scheme. His rules
and heuristics are discussed below.

**Dynamic Macroanalysis**

There have been two noteworthy analytic attacks at the systems level on the problem in which new arrivals of orders are permitted. These are by Reinitz and Jackson. The former sets up a (discrete) Markovian model of the shop and orders and suggests the use of Howard's "policy iteration method" for obtaining the optimum schedule. Reinitz' model is perfectly general, including considerations of inventory costs, customer delay (i.e., tardiness) costs, set-up costs plus any other costs which are functions of order delay as well as allowing complex routings. However, there are quite severe computational problems in pursuing his approach.

Jackson stated some general propositions regarding "job-shop like queue networks". Of importance is his "decomposition principle" in which he states the condition under which work stations may be considered to be independent. These are Poisson arrivals into the system, exponential service times, and application of the first-come, first-served sequencing rule. This applies only to serial order routings. It seems apparent, however, that his principle should hold true for other disciplines not based on processing times.

He also derived "the equilibrium joint probability distribution of queue length for a broad class of queueing-theoretical models representing multi-

---


41 Networks..., op. cit.

purpose production shops composed of special-purpose service centers...".  

By far the most prevalent and most fruitful studies of dynamic queue networks have been based on digital simulation. The first published research of this type was conducted by Baker and Djielinski, although there were apparently some concurrent efforts underway at UCLA by Sisson et al. Baker and Djielinski investigated the behavior of simple, local rules such as shortest operation and longest operation, as affected by such factors as shop size. Their major findings were that the shortest operation rule provided superior results as to mean waiting time, average order earliness, shop utilization, and average work-in-process inventory, hence its macro-effects parallel its micro-effects. They provided invaluable help for the many researchers who have followed them by noting that shop size was not significant as an experimental factor. Their model was based on simple serial orders, single channel stations and they set the pattern for a great deal of effort to follow.

The next major simulation study was that of Rowe who departed from the pattern set by Baker and Djielinski. First, he used a rather large and complex shop for his study and developed some complicated rules. He was

---

43 "Jobshop-Like......," op. cit., p. 131.


46 op. cit.
oriented towards a two-phase approach to the ultimate sequencing problem, wherein the first phase, "scheduling", established starting and release dates for the tasks and included the process of allocating "flow allowances" or "slack" to the tasks, and the second phase, "dispatching", actually accomplished the choice for next task among several waiting, based in some way upon the scheduled dates. Operationally, this type of approach is meaningful when the dispatching must be accomplished "locally": whatever sophistication is to be build into the process is done centrally, in this case, in the scheduling process. Rowe's approach is thus similar to Emery's. 47

The basic scheme he employed was this: allocate slack to orders in an inverse relationship to order value, that is, schedule higher valued orders more "tightly". Then, in the dispatching phase, employ a "schedule enforcing" rule, e.g., earliest schedule start date or minimum slack per remaining operation. This plan should result in accomplishing good performance by two criteria, first, work-in-process levels, because high value orders tend to spend less time in the shop (have lower "flow times") and second, tardiness, because later orders take precedence.

His experiments bear out this conjecture. Using a minimum slack per remaining operation rule (his "sequential rule") he obtained a successful trade-off between the two aspects of performance. In addition, Rowe studied the problem of how precise the waiting time estimates need be for purposes of scheduling. He found (as I interpret his results) that increased accuracy in waiting time estimates obtained, for example, by feeding back the results from previous simulations, was not nearly as important as the effect of using

the "variable flow allowance", (i.e., slack inverse to value). Once again, researchers who follow can derive some comfort. Rowe's experiments were factorial in level of load.

Meanwhile, experiments at a more basic level were being conducted by Conway, Johnson and Maxwell (at Cornell) who studied numerous local rules for a simple shop situation.\textsuperscript{48} This study is noteworthy not only for its useful results but also for the many insights it has provided for experimental design and procedure. It represents the first really rigorous experimentation, with attempts to come to grip with questions of sample size and initial conditions with respect to system equilibrium, factorial design, and interpretation of results. This work and later studies led to some important contributions to simulation doctrine.\textsuperscript{49}

The study was concerned with how these local rules affect completion time distributions, work-in-process levels, and shop utilization. They used an unusual mode of operation in order to obtain a wide range of performance in the latter: They held the number of orders in the shop constant, thus allowing rules which tend to increase "throughput" to undertake more orders per unit of simulated time. This is in contradistinction to fixing the order arrival rate which in turn, fixes the utilization in the long run. Their results cannot be divorced from their model, it should be noted.

\textsuperscript{48}Op. cit.

They tested thirteen basic rules which can be placed in five classes:

1. controls (random, first-come, first served);

2. due date related (earliest due-date, minimum slack);

3. processing time or number of tasks related (shortest operation, longest operation, fewest remaining tasks, most remaining tasks, shortest remaining processing time, longest remaining processing time);

4. value related rules (first-come, first-served within value class, highest value);

5. utilization related ("next critical queue", i.e., select that task whose next assignment is to a "short" queue.)

The results can be summarized as follows. With regard to utilization and mean completion time, all but three of the rules are not significantly different from random. Of these three, shortest operation and next critical queue give higher utilization and lower mean completion times, longest operation gives opposite results. With regard to work-in-process, the "continuous value" rule, i.e., highest value, behaves as a limiting rule for value class rules and hence gives the best performance. All other rules give inventory values proportional to their mean completion times. With regard to the variance of completion times, earliest due date and minimum slack provide minimum variance, shortest operation and longest operation the maximum variance. Also, it would appear from their data that one effect of increasing the load is to increase the performance differences between the two tardiness oriented rules, minimum slack and earliest due date. The more comprehensive estimate of lateness provided to the former rule seems to make a difference only at heavy load levels. (One expects, however, that since the number of tasks per order was limited to seven, more observable differences might be found with longer orders.)
Perhaps of more interest than these direct results was the experience of this group with the "equilibrium" problem. They had some problems getting their shop to "settle down" with heavy loads, and thus provided warning to those who followed of an important experimental problem.

This study was preliminary: its purpose was to isolate phenomena of interest for more comprehensive research. Conway and Maxwell chose to follow up this work with a detailed study of the virtues and drawbacks of the "shortest operation" rule. By simulation with large sample sizes (2000 orders or more) they established the following:

1. The other results of the study were remarkably insensitive to changes in shop size, load level, and degree of pattern in routing (They tested "pure job shops" and "pure flow shops").

2. The performance of the shortest operation rule (vis a vis random selection) is insensitive to errors of estimate of processing time.

3. The shortest operation rule, while optimal among the rules tested (random, first-come, first-served, shortest operation with maximum waiting time threshold, alternating shortest operation and first-come, first-served, and "next critical queue") with respect to utilization and mean flow time, has the greatest variance of flow time.

4. "Truncation", i.e., shortest operation modified by a maximum waiting time threshold, and "alternation" (with first-come, first-served) both reduce the variance but at a high price in mean flow time and utilization.

Their large sample sizes apparently alleviated their early problems with transients. Their model again was based on the "constant backlog" mode of operation.

---

50 "Network Scheduling.....", op. cit.
William Gere also conducted simulation experiments but with considerable differences in emphasis and types of conclusions. His purpose was to investigate the use of heuristics in sequencing. Heuristics, in his sense, are rules for modifying or selecting from among priority rules, based on particular circumstances. In a sense, he is looking at "complicated", sequencing rules as opposed to Conway's simple rules. More importantly, Gere was apparently not attempting to generalize about steady-state, dynamic system behavior. Many of his experiments were devoted to static macroanalysis. His results for the dynamic situations involve such small sample sizes that generalizations about steady state system behavior would be perilous. However, in that he chose sixteen small independent samples and compared ranks of his various rules, some confidence can be placed in his conclusions as to short-run transient behavior. He used six basic priority rules [slack, slack per remaining operation, ratio of slack to remaining time, excess of slack over expected remaining flow time (selecting the minimum in each of these cases), shortest operation, and shortest operation with slack ratio] alone and augmented by two heuristics used together and separately. The heuristics were:

1. "alternate operation": sequence the tasks according to the priority rule. If this sequence makes some task "critical", i.e., causes slack to become negative or fall below some threshold, place the critical task first. If this does not cause any other task to become critical, let the sequence stand; if it does, revert to the original sequence.


\[52\] These are free translations from his definitions on p. 69 et seq.
2. "look ahead" and "insert": if a (more) critical task is due to reach the machine before the priority rule selected task is completed, select the critical task; if there is a task in queue which can be completed by the time the critical task arrives, select that task.

His conclusions were as follows:

1. The selection of a priority rule for discriminating between [sic] jobs competing for time on the machines is not as important as the selection of a set of heuristics which will bolster the rule.

2. Since there is little difference in effectiveness of the priority rules after they are combined with two or more heuristics, a simple rule should be used...except for an experimental research program, complex priority rules are inadvisable.

3. The heuristics which anticipate the future process of a schedule, the alternate operation and look ahead heuristics---improve schedules significantly....

(It may be noted at this point that Gere's conclusions are precisely reversed in this work; see Chapters V and VI.)

A more recent experimental program to be reported is that conducted by Yves Nanot at UCLA. His research returned more closely to the pattern of Conway and Maxwell in that he tested relatively simple procedures with very large samples (75,000 orders in some cases). He obtained the steady-state flow-time distributions for ten rules under a variety of routing and load conditions, and also obtained results for the distributions resulting from the use of Jackson's "urgency numbers" with "due-date-like" priorities. Among his more interesting results were:

1. The emergence as a standard of the "first-in-ship, first served" rule. Nanot found this to be an excellent rule for reducing flow time variance. He suggests it as the logical system counterpart of the single station first-come, first-served rule. Its performance

---

53 Ibid, 104.
was as good as the "dynamic slack" rule (minimum of due date - remaining expected flow time - current time), with respect to the right hand tail of the flow time distribution.

2. Let $a$ and $v$ be "urgency" numbers assigned to sets of orders. Let $V_a(f)$ and $V_v(f)$ be the flow time corresponding to fractile $f$ in the cumulative flow time distribution. Then $V_a(f) - V_v(f) \rightarrow a-v$ as $f \rightarrow 1$ when sequencing with due-date like priorities.

3. If $V(f)$ is similarly defined for the first-in-system, first-served rule $V_a(f) - V(f) \rightarrow a-a$ as $f \rightarrow 1$.

If one considers the urgency numbers to be the initial lateness of the orders in the system, then the latter two results can be restated as follows: the difference between flow times at higher fractiles is approximately equal to the difference in initial lateness. That is to say, at higher fractiles, the rules tend to "make up" the initial lateness.

Nanot's results for basic rules are discussed further in the next chapter.

If Nanot's work represents an extreme solution to the sample size problem, Conway's latest study is certainly the extreme in variety of sequencing rules tested. He tested some 92 priority assignment procedures (many of them differing only in parameter settings, however). It is noteworthy that he changed from a "fixed backlog" shop to a fixed level of utilization for the studies, thus providing results more nearly comparable to Nanot's and those of this study. His sample sizes were 10,000 orders.

Conway pursued two major problem facets in this study. First, he searched for rules superior with respect to various measures of work-in-process

---

55"An Experimental Investigation...", op. cit.

56This work became known to the author when the formulation and experimentation was essentially complete. The remarkable similarities of the framework, if not the detailed experiments, are strictly coincidental.
inventory, and second, he took an exploratory peek at the tardiness problem.

With regard to the former, he found that the shortest operation rule was nearly as good as far more elaborate rules. He did find marginal superiority in his "P + XWQ" rule, which is based on a linear weighting of imminent operation time and "expected work in queue" at the next station to be visited. The expected work in queue consists of the processing times of the jobs actually in queue at the station plus that of all jobs currently being worked on which are to be routed to that queue. This is another "look ahead" rule.

He also tested a hold-off rule and found it lacking in merit.

His tardiness study was based on a criterion of minimum percent of tardy orders. Although he found minimum slack per remaining operation to be superior under reasonable due date setting procedures, the shortest operation rule provided surprisingly good performance. He also studied a weighted priority rule based on processing time and slack per operation.

A word should be said about the minimum percent tardy criterion. This is a rational criterion when there are only fixed tardiness penalties, i.e., "deadline loses". One should be aware, however, that if this criterion is used in the absence of others, it calls for abandoning (setting minimum priority on) orders which have become tardy. Hence, while the measure seems to have the ring of realism, as a serious measure of economic performance, it should be classified as of rather more academic than practical import.

Conway also oriented his work to the on-line, real-time system question; namely, is the broader date-base valuable. He found generally in the negative, to wit: "while these data do not disprove the utility and economy of such systems, they at least place on their advocates the responsibility to show that
they possess a sufficient performance advantage over a well-conceived local system". 57

Holt's Proposal

Of direct interest here is Holt's proposal for a scheduling and sequencing system. 58 Although he did not test his theory, he does define an approach that is similar to the one used here in some respects.

He suggested a two-phase approach consisting of a "global analysis" to establish release times for orders (assuming due dates as given) and schedule start dates for operations. Slack is allocated as a function of load, order value, and tardiness penalties. His dispatch rules consist of three types:

1. "Time schedule priority rule", i.e., earliest scheduled start date, to enforce the global plan.

2. "Queue cost priority rule", version 1, which is essentially a c/t rule where delay costs are derived from a quadratic approximation of the tardiness loss function and inventory costs. Wait times are assumed to be lateness invariant.

3. "Queue cost priority rule", version 2, in which delay costs incurred by other tasks in queue from selecting a given task are calculated, and the lowest cost task selected. The rule he states is based on a single channel assumption.

The similarity of Holt's framework to that of Rowe is quite close, but Holt either was unaware of Rowe's work or failed to note the similarities.

Chapter III
TOWARDS A RATIONALE FOR SEQUENCING

Introduction and Summary

In this chapter an attempt is made to draw together some of the more important elements of current knowledge of sequencing rules and to synthesize from these elements a new, generally superior sequencing system. This is done first for the classical job shop situation (classical with respect to previous research, not to the real life norm), the serial orders, single channel shop. An operable approximation to the c/t rule is proposed as a basis for sequencing and a related heuristic for "hold-off" and "sneak-in" is rationalized. For multiple channel use, a modification which recognizes channel availability time as a factor is proposed.

A digression is taken to develop definitions and notation for coping with serial-parallel order routings, and to examine the general effects of such routings on the sequencing problem. Then, a similar set of rules, that is, a c/t rule with associated heuristic modifiers for particular situations, is proposed and defended.

Some Notation (serial orders only)

The following symbols are used hereafter.

1. \( t \): present time, i.e., time at which the decision is being made.

2. \( t_i \): processing time for the \( i \)th operation.

Note: the subscript may refer to the \( i \)th operation of \( n \) in queue, or to the \( i \)th operation on an order. The context should make it clear which is meant. In the rare cases in which it is necessary notationally to distinguish between orders, a superscript will be used.
3. $c_i$: delay cost rate for the $i$th operation (at time $t$).
4. $d^k$: the due date for an order.
5. $r^k$: the release date for an order.
6. $a^k$: the allowable flow time for the order, $a^k = d^k - r^k$.
7. $q_i$: waiting time for the $i$th operation.
8. $f^k$: flow time for an order, $f^k = \sum_n t_i^k + \sum_n q_i^k$.

For a set of orders one can define a distribution of flow times, $g(f)$.

9. $T^k$: order tardiness: $T^k = \max(0, f^k - a^k)$ for an order; for a set of orders, $E(T) = \int_a^\infty (f-a) g(f) \, df$.
10. $l^k$: order lateness: $l^k = f^k - a^k$
11. $w_i$: remaining wait time, $w_i = \sum_i q_j$
11a. $\hat{w}_i$: expected remaining wait time, $\hat{w}_i = \sum_i \hat{q}_j$
12. $n_i$: normal scheduled start date for an operation.
   
   \[ n_i = d - \sum_i (t_j + q_j). \]
   
   \[ = \text{due date} - \text{expected remaining flow time}. \]
13. $u_i$: urgent scheduled start date for an operation, $u_i = d - \sum_i t_j$.
14. $s_i$: slack; $s_i = u_i - t$.

Items 12 and 13 will be recognized as handy repositories of information relevant for sequencing.
Some Operating Characteristics of Rules

Consider the problem of sequencing orders with serial routings through single channel work stations, at a fixed (mean) level of utilization, with no early shipments allowed, and a linear tardiness loss function. Suppose further that the orders are from a homogeneous population, and that due dates for all orders are set by $d^k = r^k + a$ where $a$ is a constant for all orders. This is about as simple a system problem as exists but understanding the behavior of rules in this setting will provide for later complication.

In Figure 3.1 are plotted normalized means and standard deviations of flow time distributions resulting from some of the basic rules studied by Nanot. All data are recorded as a fraction of the corresponding statistic for first-come, first served (FCFSV) in this particular run. Several points are supplied by conjectured positioning. Now, the mean and standard deviation do not give the whole picture of the flow time distribution, nor, for that matter, does the flow time distribution reveal the whole tardiness situation.

Using FCFSV as the standard, it can be seen that there are two rules which have pronounced effect on the mean flow time. The shortest operation

---

1 This section of the thesis may seem only remotely related to the work which follows it. It is included because, remote or not, it traces the thought processes which led ultimately to the system proposed.

2 Data drawn from op. cit., 55.
FIGURE 3.1: FLOW TIME MEAN AND STANDARD DIVIATION RELATIONSHIPS FOR SELECTED SEQUENCING RULES.
rule (SHTOP) provides consistently, significantly lower mean flow times; its reverse, LNGOP, yields significantly greater mean flow times and little practical interest. However, it is interesting to note the influence of LNGOP when it is included (mostly inadvertently, one suspects) in more complex rules. For example, a rule used in more than one study has been (minimum) slack divided by remaining processing time. Its use results in mean flow times greater than those obtained from FCFSV due to its small but systematic inverse correlation with processing times. Both SHTOP and LNGOP tend to have standard deviations somewhat higher than FCFSV at higher levels of utilization, but, remarkably in light of results of research by Conway et al., the standard deviation of SHTOP is essentially the same as FCFSV in the range of utilization tested by Nanot.

Parenthetically, it is interesting to review the Conway-Maxwell results in light of Nanot's. Recall that they allowed utilization to vary by holding the number of orders in the shop constant. When they noted the "disadvantage" of the SHTOP rule as being its high flow time variance (vis a vis FCFSV), they were comparing the two rules at different levels of utilization, to be sure. Furthermore, their procedure seems to have an inherent bias towards high variance in the sense of tending to create a "float" of long processing times in queues. In effect, this produces something of a last-come, first-served effect in the rules.

As a consequence of Nanot's experiments, it would seem, first of all, that SHTOP need not be precluded from consideration as a practical rule for

---

3 Rules 4 and 9, Ibid.
the tardiness problem; indeed, it becomes one of the more desirable rules; and, secondly, that any of several modifications such as the "truncation" proposed by them would make the rule difficult to surpass. The possibility of modification is discussed under "Heuristics" below.

On the other hand, it should be noted that the tendency of SHTOP towards increasing variance at high levels of utilization is pronounced. Typically, orders are routed to all or nearly all of the work stations in the shop. It appears sufficient in this case that any one station be heavily loaded to create significantly higher variance. Thus, for practical purposes, it is concluded that SHTOP must be modified by some device which would eliminate extreme tardiness.

Returning to the figure, there are three other rules which, because they sequence randomly as to processing time have essentially the same mean as FCFSV. These rules do evidence considerable differences in variance, however. The greatest variance is that of last-come, first-served (LCFSV) for obvious reasons. Now, since a random rule selects randomly both as to order of arrival and as to processing time, it ought to have a mean between FCFSV and LCFSV and a standard deviation between LNGOP and SHTOP.

Of more practical interest than the high variance rules is the earliest due date rule (EARDD). In the case in which due dates are set with a constant "a", the same sequence is obtained if a constant is added to or subtracted from the due date. In particular, if "a" is subtracted, the "due date" becomes r, the release date, and the rule is identical to Nanot's "first-in-shop, first-served" (FISFS). This rule reduces the variance of flow times since the longer an order remains in the shop, the higher its relative priority becomes.
It follows that the greater the processing time (for all operations) on an order, the less its wait time will tend to be.

It is clear enough that for minimizing mean tardiness the choice among rules discussed so far is between SHTOP and EARDD, but between them the choice is not clear. In the table below are presented some rough calculations derived from Nanot's results, using $a = \mu + \sigma$ (first-come, first-served distribution).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Shop 1</th>
<th>Shop 5</th>
<th>Shop 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>.255</td>
<td>.121</td>
<td>.176</td>
</tr>
<tr>
<td>SHTOP</td>
<td>.152</td>
<td>.069</td>
<td>.136</td>
</tr>
<tr>
<td>EARDD</td>
<td>.166</td>
<td>.083</td>
<td>.155</td>
</tr>
</tbody>
</table>

These indicate a slight preference for SHTOP at medium and low levels of utilization.

The problems of choice between reduction in mean and reduction in variance may be analyzed more abstractly. Assume that flow times are

---

4. Ibid., 131, 132, 134, 168, 170, 174, 175, 177.

5. Nanot tabulated fractiles for the distribution for each rule. Mean tardiness was obtained by multiplying the probability of an interval by its mid-point. The tail was evaluated at its left-most fractile. This is slightly biased towards SHTOP.

6. The shop designations are his, see Ibid., 45-46.
normally distributed with mean $\mu$ and standard deviation $\sigma$. Assume further that tardiness loss may be represented by a quadratic function of the following form:

$$
\mathcal{L}(f-a) = C_1 \int_a^\infty (f-a)^2 g(f) \, df + C_2 \int_a^\infty (f-a) g(f) \, df + C_3 \int_a^\infty g(f) \, df \quad (3.1)
$$

where $g(f) = \frac{1}{\sqrt{2\pi}\sigma} e^{-1/2 \left( \frac{f-\mu}{\sigma} \right)^2}$,

and $C_1$, $C_2$, $C_3$ are the squared, linear, and fixed tardiness cost coefficients, respectively. In Appendix A, the following are derived.

$$
\mathcal{L}(f-a) = C_1 \sigma^2 \left[ \int (a^2 - 1) \Phi(u) - u \phi(u) \right] + C_2 \sigma \left[ \phi(u) - u \Phi(u) \right] + C_3 \Phi(u); \quad (3.2)
$$

$$
\frac{\partial \mathcal{L}}{\partial \mu} = 2C_1 \sigma \left[ \phi(u) - u \Phi(u) \right] + C_2 \Phi(u) + C_3 \phi(u); \quad (3.3)
$$

$$
\frac{\partial \mathcal{L}}{\partial \sigma} = 2C_1 \sigma \Phi(u) + C_2 \phi(u) + C_3 \frac{u \phi(u)}{\sigma}; \quad (3.4)
$$

where $u = \frac{a-\mu}{\sigma}$, $\phi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$, $\Phi(u) = \frac{1}{\sqrt{2\pi}} \int_u^\infty \phi(t) \, dt$.

Denote the squared term in $\mathcal{L}$ by $K_1$, the linear term by $K_2$, and the constant term by $K_3$. $\mathcal{L} = K_1 + K_2 + K_3$.

It is noteworthy that:

1. $\frac{\partial K_1}{\partial \sigma} > \frac{\partial K_1}{\partial \mu}$, whenever $u$ is greater than about -.5,

   i.e., when $a-\mu > -.5 \sigma$;

2. $\frac{\partial K_2}{\partial \sigma} > \frac{\partial K_2}{\partial \mu}$, whenever $u$ is greater than about +.3,

3. $\frac{\partial K_3}{\partial \sigma} > \frac{\partial K_3}{\partial \mu}$, whenever $u$ is greater than 1.

That is to say, if only a "deadline loss" applies, ($C_1 = C_2 = 0$), then the reduction in loss from reducing $\sigma$ is greater than the reduction from reducing $\mu$, if the "deadline" is greater than $r + \mu + \sigma$. For the particular case
of interest here, that of linear loss \((C_1 = C_3 = 0; \ C_2 = 1)\), the ratio of unit loss reduction from \(\sigma\) to that from \(\mu\) is just \(\frac{\phi(u)}{\Phi(u)}\), the ratio of the density to the complementary distribution function ("right hand tail"), which is greater than 1 for due dates beyond \(R + \mu + .3\sigma\), which certainly is the relevant range.

What can be concluded from this? Just this: The partial derivatives give the "gradients" of the directions which may be pursued in designing rules. They show that, by and large, the direction of "steepest ascent" is in reduction of \(\sigma\), other things being equal; and, moreover, the more pronounced the quadratic penalty \((C_3)\), the relatively steeper is the gradient. To the degree that the penalty is worse than linear, the general conclusions are intensified. The gradients also provide easy means of assessing the effects of different loss functions on the choice of rules with particular \(\mu\) and \(\sigma\) characteristics.

To the extent that the above model is valid, the indicated direction of search is toward superior variance reduction rules, or towards variance reducing modifications of SHTOP. SHTOP is surely the ne plus ultra in reduction of mean flow time.

**The Effect of Variable Flow Allowances**

In the foregoing, it was convenient to assume constant flow allowances, \(a\), in order to discuss the tardiness effects of the flow time distributions obtained by Nanot. But the assumption is not realistic, of course. Flow allowances almost always reflect some particular characteristics of the order. The practice of quoting "standard lead times" for particular product types is
an example of flow allowance correlated with predicted flow time for the particular order, that is \( f_i \), where \( f = \sum_{i} (t_i + q_i) \). Granting that the precision of the estimate of both the processing times and certainly the queue times varies from firm to firm, it is felt that the most realistic model of due dates is the following:

\[
d = r + \hat{f} + v + e
\]

where the terms are respectively release time, predicted flow time, an allowance for variability in flow time (analogous to "safety stock" in inventory theory) and an omnibus error term. The latter includes not only errors of estimate but also such deviations from ideal that result from the due date negotiation process; it also encompasses schemes such as Rowe's for slack allocation. It is assumed that \( \sigma_e < \sigma^r \). An assumption is imbedded in this, namely that more or less standard flow time allowances for competing products are established in the market place and that the manufacturer adjusts his capacity such that his expected flow times are consistent with the standard.

The differences in the previous analysis, and conclusions drawn therefrom, resulting from this change in the basis of flow allowances is not highly important. The constant flow allowance, viewed in this framework, merely assumes that no better prediction of flow time can be made than the "grand mean".

In figure 3.2 is shown a conceptual representation of the differences in flow allowances. The large vertically oriented density function is just the overall (marginal) flow time distribution. The smaller density functions
Figure 3.2: Marginal and Conditional Flow Distributions

dangling from the line, \( a = b_o + b_1 \hat{f} \), represent the conditional distributions of flow time. The variance of the large curve reflects the variability due both to the inherent characteristics of the orders and to that of the flow through the system. The variance of the latter reflects only prediction error, due fundamentally to variability of flow through the system.

Now, if the conditional distributions are homoscedastic (have variances which are equal), the distributions of interest, \( g((f-a)|\hat{f}) \), are identical; the curve shown below is that of a homogeneous population. Homoscedasticity simply corresponds to the case in which prediction error is not systematically related to \( f \).
Figure 3.3: Conceptual Lateness Distribution

But even if the curves have different variances, all of the foregoing analysis and conclusions apply to the conditional distributions and while the conceptual basis of the discussion changes, at the level of subtlety contemplated here, the operational differences are nil.

In order to allow for plotting curves such as that shown in figure 3.3 (to which I am predisposed), the assumption will be made that the tardiness curves are homoscedastic.

It should be noted that, in assuming flow allowances of the more sophisticated form, prediction of flow times has been implicitly included as part of the problem under study.
Variance Reducing Rules

It seems highly likely that there are variance-reducing rules which are superior to EARDD. Sequencing in order of minimum slack (due date less remaining processing time, or $u_1$) would be likely to provide lower variance (though perhaps a larger mean because of negative correlation with SHTOP). In effect, the rule would consider total processing times upon release of orders rather than after the order had incurred considerable delay. Another favorite rule has been minimum slack per remaining operation (SLROP); if slack is negative, the division is not made of course. The rule has shown up well on occasion; Rowe found it best, Gere found it superior in his dynamic studies, and Conway's RAND studies confirmed this. Rules based on minimum slack are in effect earliness minimizers. But when slack is negative, the order is tardy, and the rule always selected the tardiest. Its conjectured position is shown in figure 3.1.

One can also employ rules of this same general type which include waiting time estimates as well as processing times for tasks not yet undertaken. The general earliest (normal) scheduled start date rule is of this form.

One deficiency in this type of rule derives from the fact that the tardiest (actually, least early) order is selected regardless of its effect on the tardiness of competing orders. Consider this situation

---

7 Nanot tested several such rules but apparently did not remove the division for negative slack values. He obtained very high variances as a result.

8 Gere's "modified slack", op. cit.
Assuming that these are terminal operations on each order, the tardiness resulting from a slack rule is maximum, exactly the reverse sequence is best.

The salient question is: Are there sequencing systems which can do better? Gere studied more elaborate rules but found that the two simple slack rules above were generally best. There are at least two other possibilities. The first invokes using heuristics (in Gere's sense) to override the choice of simple priority rule when the situation calls for it; "trade-offs" among competing orders may thus be explicitly examined. The second is based on using a rule which is based implicitly on "tradeoffs". This latter would be an adaptation of the "c/t rule", that is, select the maximum ratio of unit cost of delay to processing time. These two alternatives are examined in turn.

**Variance-Reducing Heuristics**

An alternative to seeking more sophisticated "same-mean, variance-reducing" rules is to search for ways to reduce the variance of SHTOP flow times while increasing the mean in a ratio of less than \( \phi(u)/\mu(u) \), this being the condition for lower tardiness loss (based on the normal assumption).

<table>
<thead>
<tr>
<th>( i )</th>
<th>slack</th>
<th>( t_i )</th>
<th>tardiness ((1, 2, 3))</th>
<th>tardiness ((3, 2, 1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-10</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>-6</td>
<td>6</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>-4</td>
<td>4</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>46</td>
<td>34</td>
</tr>
</tbody>
</table>
Conway and Maxwell experimented with two schemes for accomplishing this. One involved periodic alternation of SHTOP with FCFSV; this reverses the previously noted LCFSV aspect of SHTOP for long queues. Of course, a more powerful variance reducing rule could be used instead of FCFSV. Their other approach involved "truncation", placing an upper limit on the tolerable waiting time of a task in queue.

Alternation does not appear to be particularly useful. One reason might be some inherent start-up loss in the rules. For example, after SHTOP has been used for a period, and FCFSV is then used, until the queues are "cleaned out" it will tend to operate very much as an LNGOP rule. So the mean of the mixed rules is likely to be higher than the weighted means of the separate rules.

Truncation is intuitively more appealing, since the idea is that the SHTOP rule is allowed to govern the sequence until it creates a situation known to be undesirable. The situation is remedied and control is returned to SHTOP. Only the amount of "cure" required is used. The Cornell studies confirm the good qualities of truncation, in that even with very "weak" truncation, i.e., very high allowable waiting time thresholds, a sharp reduction in variance was achieved at a low cost in mean and utilization.

Gere's "alternate operation" heuristic is essentially a truncation rule when it is employed in conjunction with SHTOP. In this case, the truncation

\footnote{Op. cit.}

\footnote{Ibid., 293.
is performed on "slack"; when slack becomes negative on an order in queue, his rule schedules that order instead of the one with the shortest operation.

Since it is difficult to predict just how frequently truncation will take place, it is nearly impossible to assess the effect on flow time means and variance except that the latter should be reduced, of course. But eliminating the extreme tardiness cases from the SHTOP performance probably has more operational meaning than reducing its variance per se.

Gere found that his rule (in combination with "look-ahead-insert") worked somewhat more effectively when priorities were set with due-date oriented rules, e.g., minimum slack, but the differences were quite small. This would correspond to more intensive variance reduction of rules oriented towards variance-reduction already through the mechanism of recognition of the "tradeoffs". One may detect a faint underlying c/t theme in his results.

It is proposed to employ a very simple rule as representative of this class. The procedure is this: divide all tasks at a queue into two classes, tardy orders and early orders (\(t \geq u_i\) identifies a tardy order); within tardiness class, sequencing is by minimum processing time. Under an assumption of all equal tardiness costs, this corresponds to using c/t when it is known to be good, namely, for orders which are already tardy. For compactness, this rule will be referred to hereafter as TRSIO (for "truncated shortest imminent operation").
The C/T Rule, "COVERT"

The second approach to improvement is more direct. Instead of attempting to override a simple rule when it produces recognizable "mistakes", one may attempt directly to minimize the tardiness effects of delays at the local queues through use of a c/t rule, one which trades off the "cost" of the resource being allocated (processing time) against the potential "gain" (tardiness costs avoided).

For the criterion at hand, that is, linear tardiness penalties and also assuming that all orders incur penalties at the same rate, the "cost of delay" is just the incremental change in expected tardiness. As a first approximation, assume that the cost is incurred at a constant rate over the delay period being considered. The c/t rule would then be as follows: select the operation for which the ratio of incremental expected tardiness (per unit of delay) to its own processing time is maximum, i.e.,

$$\max_{i \in Q} \frac{\Delta E(T)}{t_i}.$$

The problem is to estimate the numerator. For orders which are already tardy, $\Delta E(T)$ is clearly 1, but for orders which are not yet tardy, the situation is more complicated.

First of all,

$$E(T) = \int_a^\infty (f-a) g(f) \, df.$$

Or, since $f = \sum t_i + \sum q_i$ and $a = \sum t_i + s_i$, by subtracting the common constant, $\sum t_i$, and letting $\sum q_i = w_i$,
\[ E(T) = \int_{s_i}^{\infty} (w_i - s_i) g(w_i) \, dw_i, \]

where, recall, \( w_i \) is remaining wait time and \( s_i \) is remaining slack. Expected tardiness is equal to the partial expectation of wait time less slack.

Let \( \nu \) be nonproductive time (delay). Since \( ds_i/du = 1 \),

\[
\frac{dE(T)}{d\nu} = \int_{s_i}^{\infty} \int_{s_i}^{\infty} (w_i - s_i) g(w_i) \, dw_i
= \int_{s_i}^{\infty} g(w_i) \, dw_i = \Pr(w_i > s_i)
= \Pr(\text{tardiness}),
\]

assuming that remaining wait time is independent of remaining slack. The penalty per unit time of queue delay, this says, is simply equal to the current probability of tardiness, or, expected tardiness increases proportionally to the current probability of tardiness.

Of course, since a tardiness reducing rule is being used, \( w_i \), in general will decrease as slack decreases, this is just Jackson's "dynamic priority" thesis.\(^\text{11}\) So, in reality, using this probability will tend to overstate delay costs somewhat. On the other hand, since decreasing wait time for one order necessarily increases it for others, the delay cost estimate may not be as biased as it appears at first glance.

The problem remains of how to estimate the current probability of tardiness. While this probability is a function of many measurable factors, two factors which are easily measured and obviously relevant are current slack and expected remaining wait time. And a crude approximation based on these measures would be

\(^{11}\text{Op. cit.}\)
Pr (tardiness) = Pr(w₁ ≥ s₁) = 1 - \frac{s₁}{k\hat{\omega}_1}, \ 0 < s₁ < k\hat{\omega}_1

= 0 \quad , k\hat{\omega}_1 \leq s₁

= 1 \quad , s₁ \leq 0,

where k is a constant of approximation, viz. figure 3.4.

This is represented in the following figure.

Figure 3.4: Linear Approximation to Probability of Tardiness

The dotted line reflects conjectured reality.

The basis of the approximation is quite simple. If slack is zero or negative, the probability of tardiness is by definition equal to 1. If slack, being allowable waiting time, is very much greater than the expected waiting time, then the probability is negligibly small. The precise intercept for the linear approximation to the curve is left as a matter for experimental determi-
nation.\textsuperscript{12} It may be interesting to note that TRSIO is obtained by setting \( k = 0 \), if ties are settled by SHTOP.

There is no denying that this crude, but the hypothesis is that it is better than other approaches. This approximation does have the merits of being computationally feasible, monotonic with the true probabilities, and continuous into the range \( (s_i \leq 0) \) where it is correct.

Summarizing the COVERT rule, as it shall be called hereafter, it has the following logic:\textsuperscript{13}

1. (For each operation) compute \( s_i - k\hat{\omega}_i \)

2. If \( s_i - k\hat{\omega}_i \geq 0 \), \( c_i = 0 \); if \( s_i - k\hat{\omega}_i < 0 \), go to 3

3. If \( s_i \leq 0 \), \( c_i = 1 \); if \( s_i > 0 \), go to 4

4. \( \hat{c}_i = (k\hat{\omega}_i - s_i)/k\hat{\omega}_i \)

5. \( \pi_i = \hat{c}_i/t_i \), select task by \( \max \pi_i \), ties by \( \min t_i \) (SHTOP).

Note that if all tasks are "early", i.e., \( s_i \geq kw_i \), SHTOP applies and that if all tasks are tardy, SHTOP applies.

For computational purposes, it is noteworthy that:

\[
\hat{c}_i = \frac{k\hat{\omega}_i - s_i}{k\hat{\omega}_i} = \frac{k(u_i - n_i) - (u_i - c)}{k(u_i - n_i)} = \frac{t - n_i'}{(u_i - n_i')},
\]

where \( n_i' = n_i(k-1)(u_i - n_i) \).

\textsuperscript{12} The intriguing possibility exists of performing recursive prior-to-posterior analysis to obtain more precise estimates of probabilities as a function of slack and waiting times. This is left as a subject for further study.

\textsuperscript{13} COVERT is mnemonic for "c over t". There is nothing secretive about it.
The following example may serve to clarify the mechanics of COVERT. Suppose \( k = 1 \), and the current time is 200. Note that

<table>
<thead>
<tr>
<th>( i )</th>
<th>( u_i )</th>
<th>( n_i )</th>
<th>( t_i )</th>
<th>( \hat{\pi}_i )</th>
<th>( \pi_i )</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>224</td>
<td>194</td>
<td>8</td>
<td>.2</td>
<td>.025</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>280</td>
<td>216</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>196</td>
<td>184</td>
<td>7</td>
<td>1</td>
<td>.143</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>96</td>
<td>12</td>
<td>1</td>
<td>.083</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>215</td>
<td>185</td>
<td>3</td>
<td>.5</td>
<td>.167</td>
<td>1</td>
</tr>
</tbody>
</table>

the selected task, number 5, is neither the latest, nor the "quickest" to do. The total estimated loss is minimized by selecting it, however.

In addition to the linear approximation error, the interdependency between slack and expected wait times, and the many other factors which affect the probability of tardiness, there is the important problem of approximating an inherently increasing cost with a constant cost rate. It is felt intuitively that for comparisons among tasks in queue for the short "planning horizon" involved in a sequencing decision, this is not serious. However, some heuristics to be considered search out permutations of tasks, in effect, laying out a Gantt chart, which implies a considerably longer planning period. In these situations, the approximation is improved, by "costing" the tasks at the planned time of performance. In other words, by using \( \frac{(t + t^*)}{2} \), \( t^* \) being the planned time of performance in the permutation being considered, instead of \( t \) in the calculation of the cost rate, to be multiplied by the delay, \( t^* - t \), to obtain the total imputed cost of delay.
Some Advantages of the COVERT Rule

Insofar as the approximation is good, this rule has several advantages. Most fundamental is that it does approach optimizing delay allocation at the local queue, neglecting future arrivals. At the physical level its behavior should be a compromise between that of SHTOP and (dynamic) due date rules; in fact, it corresponds to a continuously truncated SHTOP rule. It should have distinct advantages over SHTOP alone and its simply truncated versions. The information requirements for the rule (slack, expected remaining wait time, and processing time) are no greater than several previously proposed rules.

The rule is oriented explicitly towards minimizing mean tardiness. But if tardiness penalties are more complicated, having quadratic and/or constant terms, there is no conceptual difficulty in adjusting the rule. For the quadratic term, the incremental cost of delay would be a function of expected lateness; for the constant (i.e., "deadline") loss, the incremental cost would be related to the rate of change of the probability of tardiness. Thus, COVERT forms the basis for a family of rules for different criteria. Moreover, for more general models, in which inventory carrying cost can be incurred due to allowable early delivery, the cost is trivial to modify, merely by adding a constant cost per unit time to the numerator. Note also that different tardiness penalties among orders are easily accommodated by multiplying the expected tardiness \(c_i\) above by the particular penalty per unit time.

\[14\] This would increase the influence of the denominator, processing time, effecting increased SHTOP effects such as decreased flow times, which makes sense.
Some Disadvantages of the COVERT Rule

The coin has another side, unfortunately. Purely aside from the perils commonly associated with approximation, the COVERT rule has several deficiencies which derive from the rather specific conditions required for the optimality of the c/t rule. First, and perhaps most fundamental, the c/t rule is optimal for one work station with no arrivals. It is probably suboptimal for a network under dynamic load condition. A correct selection based on conditions at one work station may be incorrect because atypical conditions obtain at another work station to which orders are routed. Second, the c/t rule is optimal only for single work stations having one machine; it is known to be not optimal for multiple channels. Third, in the derivation of the rule it is assumed that all tasks are independent. This may not be highly critical for serial task routings where the interdependence is simple and largely taken care of by the estimation of "c". But it is conjectured to be important for serial-parallel routings where the relationships are complex and such pathological situations as two tasks from the same order competing in the same queue can occur.

In order to ameliorate the problems resulting from these known inadequacies of the underlying model, it is proposed to employ a number of heuristics designed to override the basic rule whenever it appears to be going grossly astray. The mode of reasoning is: 1) rank the orders by decreasing $c_i/t_i$, 2) examine the particular situation for circumstances in which sequencing by rank might be inferior, 3) apply rules which will lead to superior choices on the average. The various situations and the rules they evoke are examined forthwith.
Gere's "Look-Ahead" and "Insert"

In response to the problem of suboptimality of the rule with respect to queues considered collectively (or viewed in another way, with respect to impending arrivals), a heuristic similar to Gere's "look ahead" and "insert" is proposed. Gere's approach was to ask:

Is there a critical (i.e., late or nearly late) job due to reach this machine at some future hour, yet before the scheduled [i.e., selected] operation is [to be] completed? If so, schedule that [critical] job. Check to see the effect of this on other jobs. Either let the job remain, or replace the operation with the previously scheduled operation, depending on the resulting job lateness effected.\(^\text{15}\)

The "insert" rule then selects "the longest operation which can be fitted into the idle time gap."\(^\text{16}\) Fisher and Thompson also employed these heuristics.\(^\text{17, 18}\)

This idea has appeal. Not only does it permit judicious use of idle time, but also it should avoid decisions suboptimal with respect to those which could be made with temporal and spatial expansion of the planning horizon.

Some modifications of Gere's rules are required for use in the proposed framework. Since explicit (if approximate) delay costs are available for all tasks in queue, the decision to "hold off" a task presently in queue in favor of one yet to arrive, can be analyzed more precisely.

\(^\text{15}\) Op. cit., 68.

\(^\text{16}\) Ibid.

\(^\text{17}\) Op. cit.

\(^\text{18}\) Essentially the same rule was tested by me at Westinghouse Electric Corporation in Spring of 1960.
A Basis for "Hold-Off" and "Sneak-In" (Serial Routings)

Suppose that advance notice of arrivals of tasks into a queue is provided. Assume that the time of arrival of each task not already present is known with certainty. For each task present or coming there are three numbers of direct interest: \( t_i \), its processing time; \( c_i \), its delay cost rate; and \( h_i \), its time until arrival (\( h \) is mnemonic for "hiatus")). Assume furthermore that, except for those tasks for which advance notice has been given, there will be no other arrivals (the standard \( n \)-job assumption). Order the tasks arbitrarily. Let \( T_{j-1} \) be the earliest completion time of the \( j-1 \)st task under the ordering. Then, the delay cost for the \( j \)th task is:

\[
C(j) = \max \left\{ \sum_{i=1}^{j-1} t_i - \sum_{i=1}^{j-1} (h_i - T_{i-1}) - h_j, 0 \right\} c_j. \tag{3.5}
\]

More importantly, for any two tasks adjacent in sequence, \( j \) and \( j+1 \), consider interchanging them in sequence. The variable costs for the two sequences are:

\[
V(j, j+1) = (t_j + h_j - h_{j+1}) c_{j+1} + (h_j - T_{j-1}) \sum_{i=j+2}^{n} c_i
\]

and

\[
V(j+1, j) = (t_{j+1} + h_{j+1} - h_j) c_j + (h_{j+1} - T_{j-1}) \sum_{i=j+2}^{n} c_i
\]

\(^{19}\) The change in terminology from "look-ahead" and "insert" is made for a number of reasons. First, hold-off and sneak-in have historical roots in my previous research and I am emotionally unwilling to abandon them. Second, I think they are more descriptive of the process. And third, "look ahead" can be reserved for the logic of considering the next or subsequent assignments of tasks in queue, that is, looking ahead both in time and in routing, for which process it is more descriptive.
neglecting the trivial cases in which

\[ h_{j+1} \geq t_j + h_j \quad \text{or} \quad h_j \geq t_{j+1} + h_{j+1} \]

and also neglecting in the term \( \sum_{i=j+2}^{n} c_i \) the more important fact that there may be "hiatuses" in the tasks which follow the second of the pair considered such that the current hiatus would be "absorbed". Given this equation, to say that

\[ V(j, j+1) < V(j+1, j) \]

implies that

\[ t_j c_{j+1} + h_j \sum_{i=j}^{n} c_i < t_{j+1} c_j + h_{j+1} \sum_{i=j}^{n} c_i. \]  \hspace{1cm} (3.6)

Following this line of reasoning through a number of pairwise interchanges, the first task in sequence should be that for which

\[ t_i c_j + h_i \sum_{i=1}^{n} c_i \text{ is minimum, } j \neq i. \]

This does not consider "sneak-in" possibilities. If the decision is made to hold-off for \( h \) time units, and there exist tasks with processing times less than \( h \), then a lower total cost is incurred.

In general the cost saving from sneaking in the \( j \)th item is

\[ c_j T_{j-1} + t_j \sum_{i=j+1}^{n} c_i, \]

again assuming no gaps.

Given the problems of gaps, the general difficulty of sneak-in analysis, and the practical approximation problem with a somewhat extended planning
horizon (i.e., the accurate estimation of \( c_i \)), the following heuristic was developed:

1. Restrict arrival notices to tasks which are already tardy (thus simplifying the programming to a great extent);
2. If the number of tasks actually in queue is \( n \) or greater, do not consider holding-off (this reflects the fact that the idle time will ultimately delay orders not arrived at this time);
3. Order the tasks by decreasing \( c_i/(t_i + h_i) \) ratio;
4. If the first task in the ordering has \( h_i = 0 \), select that task;
5. Otherwise, compute the total delay costs with and without holding off for the arrival of the first ranked order, considering all sneak-in possibilities, and gaps.

Elaborating, if a potentially interesting hold-off situation occurs (step 5), delay costs for the hold-off are calculated by ascertaining the predicted starting time \( t^*_i \) for each task in queue. If a sneak-in is possible it is made (sneak-ins are considered in order of decreasing rank). If there is more than one arriving order, its \( t^*_i \) is calculated as the maximum of the finish time of the previous task or its arrival time. The cost rate is calculated as \( t_c = (t + t^*_i)/2 \) and multiplied by the delay of \( t^*_i - t \). The costs are then obtained for the highest ranked, physically present, order in first position. If the first alternative yields a lower total cost, it is selected. The threshold, \( n \), for the queue length is an experimental parameter.

**Operating Characteristics**

The procedure described, if compared with no rule at all, i.e., \( c_i/t_i \) sequencing, almost never holds off when it should not. Except for
the problem of delay of unannounced arrivals (and this is a major exception), it will never produce a sequence worse than that generated by COVERT.

The rule protects against "chains" of idle time. A chain occurs when a hold-off is performed, then, when the selected task arrives, another hold-off is triggered, and so on. A second hold-off will never be performed for tasks considered at the time of the initial decision. The selected task, 1, has

\[ t_1 + h_1 \leq t_j + h_j \text{ for any } j \text{ for which } h_j > 0. \]

It arrives after \( h_1 \). Its priority will then be \( 1/t_1 \); that of \( j \) will be

\[ 1/(t_j + (h_j - h_1)), \]

since only if \( h_j > h_1 \) will another hold-off be possible. But it follows from the original selection that

\[ t_1 \leq t_j + (h_j - h_1), \]

and ties are resolved in favor of the present task. Now, a limited chain may occur because of subsequent notification of higher priority tasks. However, this is likely to be rare because of the built-in biasses in the rule plus the limitation of only critical orders for advance notice.

Finally, the mechanism of advance notice requires description. It is a simple process. Whenever a "critical" task is placed on a machine, a "dummy" task, with time until arrival is placed in the queue of its next work station. Under the assumptions of the model, \( h_1 \) is known with certainty.
Multiple Channel Work Stations

The c/t rule is not, in general, optimal for multiple channel work stations. The multiple channel problem is a complicated one even for no arrivals. Rothkopf has obtained a dynamic programming formulation for the static case of relevance here, that of delayed channel availability (i.e., where not all channels become available simultaneously), but even assuming the static case to be relevant the computations are too lengthy to be considered.\(^\text{20}\)

A modification to c./t._i is proposed for testing. Let _i, _i=1, 2, ..., _n be the task index and let _j, _j=1, 2, ..., _n be the machine index for a work station. Assume all tasks to be present. Suppose one machine, _k, has become available. Denote by _e the time when the next machine (not _k) will become available, and let \(\tau_i = \min(t_i, e)\). Now \(\tau_i\) is just the minimum delay for the next task to be processed. The rule proposed is to select by \(\max \frac{c_i}{\tau_i}, \tau_i > 0\), \(\max c_i\) otherwise, ties settled by SHTOP and then task index (randomly).

The properties of this rule are not perfectly obvious. However, the following can be ventured:

1. The rule is optimal if _n=2 (certain in the no arrivals case)
2. As the queue length increases, the merit of the rule decreases relative to c./t._i sequences (conjectured);

\(^{20}\text{Ibid.}\)
3. For long channel delays (high e), the rule behaves as $c_i/t_i$; for short delays as $\max c_i$ (tautological);

4. The rule would tend to alternate between these two, or more generally for a $m$ channel station, would tend to behave as $\max c_i/t_i$ about $1/m^{\text{th}}$ of the time, as $\max c_i$ for the remainder (heroically conjectural).

Even assuming the last of these is true, it is not known whether this is to be desired. In defense of the rule, it can be stated that it has performed moderately well in pencil and paper simulations which also support conjectured property (4 above). As to property 2, a test will be made with a switch to $c_i/t_i$ when $n > k$, with $k$ as an experimental parameter.

There is another COVERT modification which, while it is less defensible of theoretical grounds, is interesting enough to test. The problem in the multiple channel situation is that the processing time of an individual task is not a perfect measure of the delay it causes other tasks. A mechanical way to reduce the weighting placed on $t_i$ is simply to add a constant to $t_i$. And, in fact, a plausible choice for the constant is the mean processing time for the group times the number of machines in the group less one, i.e.

$$\pi_i = \frac{c_i}{t_i + (m-1) \mu_t}$$

**Hold-Off Rule for Multiple Channels**

The hold-off rule for multiple channels proceeds precisely as for single channels except that $T$ may be used instead of $t$ in the initial ranking. Of course, in computing the $t_c$ (time a task is planned to start) in the evaluation, the availability of all machines must be considered. In effect, a static simulation is performed to obtain these times (very like chess programs).
Heuristics for Serial-Parallel Routings

The problem of sequencing tasks on multiple component orders is complicated. Consider for a moment some of its basic features. First of all, total flow time for an order no longer can be simply subdivided into order processing time and order queue time. The additional phenomenon of "staging" has become important. For example, a simple routing for a four component order is shown below. The four components of the order are routed independently before being assembled together in one final operation. The flow time of the order would be the longest flow time of the four independent segments plus the waiting and processing time for the final joint operation.

( P is a pseudo operation requiring no time, but requiring completion of both preceding tasks for its performance.)

Figure 3.5: Example routing for a multiple component order.
Neglecting problems of the terminal operation itself, it can be seen that variance in flow times of individual components has taken on increasing importance. For example, if the probability of lateness (late arrival for assembly in this case) of one component is \( p \), the probability of late start for the terminal operation is \( 1 - (1-p)^n \). Another way of viewing this is that staging delay is preventable just as waiting is. The amount of staging delay depends on how well the parallel routed components are kept "in phase" or "on schedule". Thus, one may be willing to trade some additional wait time to obtain lower component flow time variance, other things being equal. And very fundamentally, there is no premium for very short flow times for a component unless components routed in parallel also have short flow times.

**Serial-Parallel Orders: Formal Description**

A digression is made here to provide a formal description and discussion of the internal interrelationships of multiple component orders as they are construed in this study. This description is set forth with two purposes in mind: first, to provide unambiguous definitions of those attributes of these orders which are relevant to sequencing decisions; and second, to suggest the information processing system required to provide current attribute values to the decision system. Certain assumptions are made about multiple component orders; these are identified as such.

21 The routing portrayed in figure 3.5, that is, independent operations on all components until final assembly, while of interest, is not the type of routing to be dealt with experimentally. For one example of the type worked with see figure 3.6. William Maxwell of Cornell and RAND in a personal communication informs me that he is studying just the type of routing shown in figure 3.5.
An order consists of \( m \) components on which operations are performed. An operation is denoted (initially) by a subscript \( i \), and is specified by the machine group performing the operation \( g_i \) and the time required to perform it \( t_i \). The sequence of operations within an order can be stated in the form of relationships called directly precedes which are denoted by "\(<\)". To say that operation \( i \) directly precedes operation \( j \) or that operation \( j \) is directly preceded by operation \( i \) (\( i < j \)) means that operation \( i \) must be completed before operation \( j \) can be started, and that no other operation on the order intervenes.\(^{22}\) The relationship is required to be logically consistent, of course, permitting no cycles, e.g., \( i < j, j < k, k < i \).

The following types of operations are distinguished.

1. **Terminal operation** (\( f \)): the final task on the order; when \( f \) is completed, the order is completed. A terminal operation directly precedes no other operation. An order has exactly one terminal operation (by assumption).

2. **Initial operation** (\( s \)): a pseudo-operation (i.e., one having no real-life counterpart, requiring no time and no real facilities) corresponding to the act of release (raw material arrival) for an order component. No other operation directly precedes an initial operation.

3. **Assembly operation** (\( l \)): a pseudo-operation which is directly preceded by exactly two other operations (by assumption).

4. **Serial operation** (\( i, i \neq l, i \neq s, i \neq f \)): an operation on one or more components of the order. It is directly preceded by exactly one other operation.

**Assumption:** Except for \( f \), an operation directly precedes exactly one operation. Note that this precludes disassemblies.

It is asserted that this framework is perfectly general (except for disassemblies). For instance, an operation which calls for assembly of

\(^{22}\text{Note the change of convention: the subscript is now used to distinguish among operations on an order as opposed to operations on different orders in a queue.}\)
three or more components can be represented by two assembly operations (as defined above) followed by a serial operation with the work station and processing time required assigned to it. The pseudo-operations are used purely for logical and computational convenience. Figure 3.6 shows a schematic representation of a typical order.

It is useful to define the following sets of operations:

1. **Route**: any finite, non-zero number of operations connected by direct precedes relationships; hence, a generic term.

2. **Path**: a route which has as its first member an initial operation and its last member the terminal operation. There are $m$ paths per order, each corresponding to the route followed by a component. Since components, once assembled, follow identical routes, paths are not mutually exclusive (all paths include the terminal operation).

3. **Segment**: a route having as its first member either an initial operation or an assembly operation, which contains no assembly operation elsewhere in the route, but which contains all possible serial or terminal operations given this constraint. There are exactly $2m - 1$ segments per order, $m$ originating from the initial routes of the components and $m - 1$ originating with subsequent assembly operations.

Now, a path may be partitioned (subdivided into mutually exclusive, exhaustive subsets) into segments. This follows from the fact that the first segment can contain the initial operation and all subsequent serial operations, the succeeding segments each assembly and subsequent serial operations until the final segment contains as its last element the terminal operation. Likewise the order routing can be defined as the set of all ordered operations, and can be partitioned into segments.

The segment is therefore useful as the basic logical element in multiple component order routings. It represents the portion of the order routing wherein the components follow independent routing.
The shaded operations form a path.

FIGURE 3.6: TYPICAL SERIAL-PARALLEL ROUTING
Finally, a relationship which is analogous to directly precede for tasks may be defined for segments. A segment $\alpha$ is said to directly precede segment $\beta$ if segment $\alpha$ must be completed (i.e., its last operation completed) before segment $\beta$ (i.e., its first operation) can be started, and no other segment intervenes.

A full notation for tasks may now be given. The subscript $i\alpha$ means the $i$th task on segment $\alpha$; $s\alpha$ is reserved for initial operations so $l\alpha$ will always denote assembly operations; $f\alpha$ is the terminal operation.

When an order is in process its elements may occupy several states, but for computational purposes only two are important: active and inactive. An operation $i\alpha$ is active if all operations which directly precede it have been completed, but $i\alpha$ itself has not been completed. An operation which is not active is inactive. An active operation is in queue or in process. Note that an operation in staging, that is an assembly operation for which one but not both of its directly preceding operations are finished, is inactive.

It is clear intuitively and formally true that there can be at most one active operation per path (otherwise the directly precedes relationship is inconsistent). Thus there can be at most $m$ active operations per order. Now, if an active segment is defined as one containing an active operation, then there are at most $m$ active segments as well. Since a segment becomes inactive, never to become active again, at the completion of its last

---

23 This is quite useful computationally; it defines the number of records on "slack" (see below) which have to be maintained.
operation, and since two active segments are replaced by one in execution of an assembly, the number of active segments is exactly \( m-k-h \), where \( k \) is the number of completed assembly operations and \( h \) is the number of operations (and hence segments) in staging.

Let \( A \) be a set of sets: namely, the set of active segments. The segments in \( A \) contain all operations for which sequencing decisions may be required, and, it follows, all those for which detailed status information may be required.

**Schedules and Computation of Slack**

Consider now the problem of measuring the lateness of an order and the relative lateness of components within an order. Define \( \hat{q}_{i\alpha} \) to be the expected queueing or staging delay, as appropriate, before operation \( i \) on segment \( \alpha \). (Since these times are a function of the sequencing system, assume that these are for first-come, first-served sequencing.) For each operation which is not the last operation on a segment, define:

\[
 n_{i\alpha} = n_{i+1,\alpha} - \hat{q}_{i+1,\alpha} - t_{i\alpha};
\]

for the last, but not terminal operations,

\[
 n_{i\alpha} = n_{1\gamma} - \hat{q}_{1\gamma} - t_{i\alpha},
\]

where \( \gamma \) is the segment directly preceded by \( \alpha \); and define for the terminal operation

\[
 n_{f\alpha} = d - t_{f\alpha},
\]

\( n_{i\alpha} \) is the normal scheduled start date for the operation.
Preceding in a similar way, define for "interior" operations of a segment $\alpha$

$$u_{i\alpha} = u_{i+1,\alpha} - t_{i\alpha}$$

for "last" but not terminal operations

$$u_{i\alpha} = u_{i\beta} - t_{i\alpha}$$

and $u_{f\alpha} = d - t_{f\alpha}$ for the terminal operation. $u_{i\alpha}$ is the urgent scheduled start date for the operation; an operation not started by its urgent start date signifies an order which is tardy.

Note the following identities:

1. $\hat{w}_{i\alpha} = u_{i\alpha} - n_{i\alpha}$, the expected remaining waiting time for any operation

2. $s_{i\alpha} = u_{i\alpha} - t = u_{i+1,\alpha} - 0_{i\alpha}$, the "slack", or allowable remaining wait time for any active, waiting operation

where $i+1,\alpha$ is taken generically as the task which $i$ directly precedes.

The general principles for multiple component orders may now be developed. Tardiness penalties are assessed against orders, therefore the relevant probability of tardiness is that of the order as a whole. The probability of tardiness for the order is derived from that component (and hence, active segment) likeliest to be tardy. Therefore

$$c_0 = \max_{\alpha \in \mathcal{A}} (c_{\alpha})$$

where $c_{\alpha}$ is of course derived from its active operation.
Segment slack is defined as the delay tolerable by the active task until the risk of tardiness penalties is directly increased. Since \( \frac{\partial c_\alpha}{\partial \mathcal{U}} = \frac{1}{\mu_\alpha} \),

\[
\frac{s_\alpha}{w_\alpha} + c_\alpha = c_0 \quad \text{or} \quad s_\alpha = (c_0 - c_\alpha)w_\alpha, \quad 0 < c_0 < 1.
\]

For orders which are already tardy, \( c_0 = 1 \), and does not change. The procedure is to define

\[
T_o = \max_{\alpha \in \mathcal{A}} (T_\alpha) \quad \text{where} \quad T_\alpha = t - u_\alpha
\]

\[
s_\alpha = T_o - T_\alpha
\]

For orders which are early, \( c_0 = 0 \). To obtain the segment slack simply compute, \( n_i - t \), which is the amount of delay tolerable without \( c_0 \) becoming \( > 0 \).

The computation of these quantities which are changing constantly is not simple; the problem is discussed in detail in Appendix C.

Hereafter, the discussion will return to the problem of selecting a task from a queue. The notations \( c_i \) and \( s_i \) will be used, \( c_i \) being understood now to be equal to \( c_0 \), the order delay costs and \( s_i \) being understood to be equal to the task's segment slack. Subscripts will now be used exclusively to distinguish among tasks in the queue.

The Significance of Segment Slack

In summary, segment slack is an attribute of an active segment of an order; it represents the amount of delay that the active task in the segment can tolerate without delaying the order. Now, tardiness penalties being assessed against the order, segment slack represents the delay tolerable
until the risk of tardiness penalties is directly increased. Thus, in considering the sequencing of tasks at a work station the delay effects are removed one stage from the order; if the task is on the "critical path" \((s = 0)\), delaying it will delay the order; if it is not, it may be delayed for \(s\) units of time before the order is affected.

**General Scheme for Sequencing**

The proposed procedure is this. Use maximum \(c_i/t_i\) to obtain an initial ranking of the tasks in queue (\(c\) is set equal to \(c_0\) for this purpose, recall). Use heuristics to overrule this ranking for tasks having sufficient slack to permit costless delay. In general, a task \(i\) will not be selected, regardless of its \(c_i/t_i\), if its slack is sufficient to permit another task \(j\), with lower ratio and less slack, to precede \(i\) without adding to \(i\)'s order's potential tardiness.

**Hold-Off Rules**

There is one simplification in handling multiple component orders that compensates partially for the many complications introduced, this being the assessment of hold-offs. The hiatuses (times until arrival) for arriving tasks are directly equivalent to slack in the sense that they represent the time until delays become damaging. That is, one may merely add \(h_i\) to \(s_i\) to obtain the time until delay costs apply. But for purposes of allocating machine capacity, \(h_i\) is additive to \(t_i\), that is \(h_i + t_i\) is the length of time other tasks are delayed, neglecting sneak-ins. The general procedure for hold-offs is then simply to rank orders by \(\max c_i/(t_i + h_i)\) and to consider slack as \(s_i + h_i\) in the heuristics.
**Single Channel Work Stations**

The procedure for single channel work stations is as follows:

1. As before, if there are n or more tasks in queue, do not consider hold-off.

2. Rank by \( \frac{c_i}{(t_i + h_i)} \).

3. Select the top ranked four tasks subject to the condition that at least one task must be present (\( h=0 \)).

4. For each of the four tasks, compute an "effective cost time" of \( t_e = t + s_i \). This is the point in time after which delay penalties are assessed.

5. Compute the total delay costs for each of the permutations of the four tasks, where, for the particular sequence, the delay cost for any one task is \( \max\{0, t_c - t_e\} (t_e + t_c)/2 \).

6. Select the first task in the lowest cost permutation (if \( h>0 \), hold off).

If four or more tasks are in queue, the 24 permutations are evaluated. Naturally, if fewer than four are present, only the necessary permutations are considered.

**Multiple Channel Machine Groups**

For multiple channel machine groups, the basic ranking is the same, but the rule provides an option which attempts to fit the tasks to the particular pattern of machine availabilities. Specifically, the procedure is (machine k having become free):

1. Rank tasks by \( \frac{c_i}{(t_i + h_i)} \)

2. Search the machines for that having the latest availability time not making the selected task late, i.e., search for \( f_{j} \) given that \( f_{j} \leq s_i + h_i \quad j \neq k \)

\( (f_{j} \) is the time until machine j becomes free.)
3. If an alternate machine is found, discard the task from consideration, proceed to the next ranked task and repeat step 2 (with the previously allocated machine updated as to its availability). Repeat until no alternative machine can be found.

4. Proceed as before in evaluating permutations of tasks not discarded.

The alternate machine option is provided in two versions, in the first, the alternate search is only provided for tasks not yet arrived, i.e., potential hold-offs, in the other, the search is made for all tasks in rank order.
Chapter IV
STATEMENT OF HYPOTHESES AND EXPERIMENTAL PLAN

Introduction and Summary

In this chapter are stated the specific hypotheses to be tested. These all derive from the fundamental normative hypothesis, namely, that the sequencing system proposed in the previous chapter is effective relative to other systems. The simulation programs are described briefly here with emphasis on some of the more difficult problems encountered, most notably: compact representation of multiple component order routings, computation of "dynamic" critical path information (i.e., slack), and efficient realization of the rules and heuristics themselves. The experimental plan designed to test the hypotheses is outlined. This includes choice of the competing sequencing systems, the basic conditions of the test as to orders and shop configurations, and a discussion of the "tactical" problems of the experiment as defined by Conway.¹ These problems are selection of starting conditions, questions of "equilibrium", measurement, sample size, and test procedures.

The Rules to be Tested

The general hypothesis to be tested is that the COVERT rule, augmented by heuristics for particular situations, is effective. That is to say, the rule will provide lower mean tardiness than some other rules previously or presently proposed, in the long run (i.e., in the "steady state") and under fairly general

conditions of load, shop configuration, and order routings.

The rules which have been selected for testing are:

1. First-come, first-served (FCFSV)
2. Shortest imminent operation (SHTOP)
3. Truncated shortest imminent operation (TRSIO)
4. Earliest scheduled start date (EARSD)
5. Minimum slack per remaining operation (SLROP)
6. "Gere's Best", which is SLROP augmented by his "alternate operation" and "look-ahead and insert" heuristics. (GERES)
7. Maximum \( c_i / t_i \) as defined in the previous chapter (COVERT)
8. COVERT augmented by the appropriate "pattern" heuristic, e.g., modifications for multiple channel work stations and/or serial-parallel routings (CTPLS)
9. CTPLS augmented by the appropriate "hold-off" and "sneak-in" heuristics (WORKS)

The first six of these are defined forthwith.

FCFSV and SHTOP require no elucidation except that in case of ties the selection is random. TRSIO was introduced in Chapter III. It is of the simplest type: tasks are divided into two mutually exclusive classes, early and tardy, \((u_i < t \text{ defines the latter})\); within classes, selection is by SHTOP, ties being settled randomly. This is a naive COVERT rule in that it does select within the tardy class by max \( c_i / t_i \) (under the assumptions of the model). It has the merit of utter simplicity and serves as an appropriate standard of comparison for the more complicated rules. It should be considered as a member of the COVERT family.

The earliest scheduled start date rule is also very simple. The start
date is, as defined previously, the "normal" scheduled date. It is

\[ n_i = d - \sum_{j=i}^{n} (t_j + q_j) \]

or due date less expected remaining flow time. The rule is of interest because it is in common use.\(^2\)

The SLROP rule is included because it was found relatively effective by Row, Gere, and Conway and is significantly different in structure from EARSD to warrant separate testing. Its rationale is simple, select the task which has the least allowable non-productive time per remaining operation; if some tasks are tardy, select the tardiest. Gere's rules are to be used for the single machine, serial routings case, but only for that case. Since he did not specify any modifications of his rules for multiple channel cases or for multiple component orders, it would hardly be fair to compare beyond the scope of his study.

**Statement of Hypotheses**

The general hypothesis is broken down into a set of more specific hypotheses. These are:

I. The COVERT rule is more effective than any other rule to be tested for the classical model (single channel work stations, serial order routings).

II. The hold-off rule (always used with sneak-in) is effective in this case.

III. The conclusion holds regardless of load level (within reasonable operating limits).

\(^2\) In particular, it is in use in the shop used as a basis for the model.
IV. The conclusion holds regardless of the "tightness" of flow allowances.

V. The multiple channel modification (CTPLS using "τ") is effective relative to COVERT.
   a. two channel stations
   b. four channel stations

VI. The "additive" rule is effective for multiple channel situations.

VII. Hold-off is an effective adjunct to the otherwise most effective rule.

VIII. The full rule (WORKS) is effective for serial order routings for general (as to channel arrangement) shop configurations.

IX. The pattern heuristic for multiple component orders (CTPLS) is effective relative to COVERT for three-component orders (at a typical load level).

X. The pattern heuristic for multiple channel machine groups as well as for multiple component orders is effective relative to COVERT for three-component orders and two-channel stations.

XI. Returning to single channel machine groups, CTPLS is effective for six-component orders.

XII. The more complex the order routings (i.e., the more components), the more effective is CTPLS.

XIII. For the cases implied in hypotheses IX-XII, hold-off is an effective adjunct to CTPLS.
The plan implied by the specific hypotheses is the testing of the COVERT system, feature-by-feature. The plan may be diagrammed thus:

Figure 4.1: Schematic representation of the experimental plan.
The Simulation Program

It is no great accomplishment nowadays to have written a job shop simulation program. At least it is not pioneering; major programs have been produced by IBM, Cornell, Nanot, Gere and others. Modified versions of the original IBM 704 Job Shop Simulator are rather common in industry. In fact, many features of the original 704 Simulator can be seen in the new "general" simulation languages, GPSS of IBM, and SIMSCRIPT of the California Analysis Corporation. No new trails are blazed here.

Why not make use of one of the many others? Nearly all are compatible with local equipment (IBM 7094). The answer is this: All are too slow and wasteful of memory to permit experiments of the scope proposed here. As will become evident when the experimental plan is detailed, long runs are planned. It has become clear, it is assumed, that the sequencing system itself and the internal information system required to operate it are complex. While the decisions take perhaps a few milliseconds each for their implementation, they do tend to add up in runs requiring 50,000 or so. Yet runs of such length are required to establish "steady state" conditions with high

---

3 IBM Corporation, op. cit.


5 I have personal knowledge of programs at Allis-Chalmers, General Electric, United Aircraft, and Westinghouse Electric.


8 Perfectly feasible on an on-line, real-time computer.
confidence, and the argument for the rules themselves has already been declined.

Nanot used SIMSCRIPT and 125 hours of 7094 time for his very simple shops and very long runs. 9 I have used "FAP", (an acronym for Fortran Assembly Program) a symbolic language, in an attempt to wring very nearly the last drop of efficiency from the computer. To this purpose, effort has been made to utilize the utmost power of the computer, including full use of all index registers, attempts to take advantage of "look ahead" (IBM's not Gere's) in the main processing loop, avoidance of "floating point" arithmetic, and rather extensive use of threaded and push-down lists to accomplish both storage and access efficiencies. 10

One problem requiring some programming effort is that of capturing compactly the complex precedence relationships of tasks on multiple component orders. Another is affording a reasonable method for performing the "dynamic critical path" updating calculations. The solutions to these problems appear in appendix C. The conceptual design problem on critical path updating is discussed below.

Organization of the Program 11

There are actually two separate programs; one for generating the input for

---


10 All of these terms are defined and very lucially discussed in R. S. Ledley, Programming and Utilizing Digital Computers, (New York, McGraw-Hill, 1962).

11 A full description and documentation of this set of programs, together with other features currently being developed, e.g., alternate routing and machine substitution, is the subject of a Project MAC technical report scheduled for release in August, 1965.
the simulation, and the simulation proper. These are discussed in turn.

A. The Input Generator

This program embodies a straight-forward procedure for generating orders via Monte Carlo from parameters supplied by the experimenter. It also includes provisions for generating routing "trees", i.e., serial-parallel routings, which is not so straight-forward. Among other choices supplied the experimenter are:

1. Number of orders.
2. Number of "non-data" orders (no tardiness statistics collected).
3. Order interarrival times (Erlang k; pick the k and mean).
4. Initial "earliness" (tabular distribution).
5. Components per order (tabular distribution).
6. Transition matrix.
7. Number of operations per segment (tabular distribution).
8. Processing time distributions (Erlang k; pick the k and mean).
9. Mean processing times for machine groups.
10. Waiting time allowance for machine groups (used for scheduling).

Using these parameters, the program generates individual tasks for each segment, combines the segments for the full order routings, calculates due dates, then "backdates" to obtain the "normal" and "urgent" start dates for each task in the routing. (It will also calculate these dates for routings supplied by the experimenter).

12 The programming and debugging were performed mostly by Ted W. Schwenke and Thomas J. R. Johnson. The blame for the design is largely mine.
It further "packs" the generated order routings (in release time order) into the format required for the simulation itself. It produces a load analysis time series by machine group, as well as other statistics for checking the Monte Carlo process, as a by-product of the scheduling routine.

Space and efficiency are less critical considerations for this program; one input tape (or simulated tape on disk) supplies several simulation runs. Therefore, Fortran II was used.

B. The Main Program

The main program simulates the flow of orders through a specified shop (each of the shop configurations used actually uses a different program), using whatever decision rule is supplied to it for sequence choices, and collecting a bare minimum of statistics on the "permanent entities" e.g., machines and "temporary entities",\textsuperscript{13} jobs, as simulated (and, unfortunately, real) time passes. Its inputs are, of course, the previously prepared orders and a few parameters to direct the run.

Provision is made for "dumping" shop status and current backlog at the planned termination of a run, in order to facilitate continuation in case of unexpectedly lengthy runs, recalcitrant machine operators, evident transients in the output and the like. In particular, the experimenter specifies a desired run length in terms of length of simulated time or in terms of number of "data" orders, (one from which statistics are collected) completed. The latter mode is used throughout the experiments described here.

\textsuperscript{13}The quoted terms are derived from SIMSCRIPT, Markowitz, et. al., op. cit.
There are several outputs supplied by the program, at varying intervals of time. The most frequent output is a one line "snapshot" of the number of tasks in queue for each machine group. These are produced "daily", a "day" being defined by the experimenter, and are useful for assessing the stability of the shop. At the option of the experimenter, periodic statistics on waiting and staging times may be reported. Data orders are designated as belonging to data blocks, each containing 32 orders (in later runs 64 orders). As each order is completed, its tardiness is added to that accumulated for the block. At the termination of measurement, cumulative mean wait times, mean tardiness for each block, a grand mean tardiness for all data orders and a histogram of order lateness for all data orders are printed out.

The gross internal logic of the program is shown in the flow chart in figure 4.2. After reading in the initial load (e.g., from previous runs), the basic control loop is entered. The inception of the loop is always the "What happens next?" question. There are two types of events corresponding to shop activity. One of these is a completed task (which is close kin to the release of a component from a previously released order which event shares some of the same logic). The machine is set to "idle" and the succeeding operation on the order is placed in its queue (REMOV). If the completed operation was the terminal operation, the tardiness of the order is computed and appropriate statistics are updated. When all completed tasks have been treated, transfer is made to the ASSIGN routine in which selections of next tasks are made for all idle machines (for which tasks exist in queue).

The SELECT subroutine contains whatever rule is being tested including all heuristics. It returns to ASSIGN with a selected task or an explicit decision to hold-off.
INTLZ: Read in run-controlling parameters, initialize all accounts, read in initial shop status (unless empty).

TIMER: Ascertain time of next event and set $T$ to that time.

RELES: If an order release is next, read it in and place it in queue. If there is an idle machine in the group of first assignment, place the job on the machine. (For multiple component orders, this applies to each branch of the order; when an order is read in, branches not scheduled for release are placed in a pseudo-queue which is polled for earliest release by TIMER).

REMOV: If a task has been completed, remove it from the machine, set the machine idle. If the task was terminal (and the order a data order) compute tardiness, update data block count and lateness histogram. Otherwise, place the next eligible task in queue. Do for all completed tasks.

Figure 4.2: Cross flow chart for the simulation program (continued on next page).
**ASIGN:** If there is an idle machine and a task in queue, call SELECT to obtain the chosen task; place it on the machine, remove it from queue, update wait time accounts. Do until there are no idle machines with tasks waiting (unless a hold-off has been indicated).

**EODAY:** If \( T = \text{end of day} \), print out number of tasks in queue for each machine group and check for \( T = \text{end of period} \). If yes, print out number of tasks and mean wait times for each group. Compute new end of day and/or period. Otherwise, go on.

**EORUN:** If \( T = \text{end of run} \), or if last data order has been completed, print out cumulative wait time statistics, block and grand mean tardiness and lateness histogram. Otherwise, go to LOOP.

**ONNON:** If experimenter desires, write tape containing order file of all current WIP, all queues and machines and certain other status accounts. Correct all time accounts relative to end of run. Otherwise stop.

**STOP**

*Figure 4.2: Gross flow chart for the simulation program, continued.*
Selected tasks are placed on their machines then. If a hold-off heuristic is being used, a selected critical task also cues the notification of its next assigned machine group of its impending arrival. Also, following the choice of a task for assignment, when all idle machines in a group are exhausted, the active segments are updated for each task not selected and hence delayed for a known time. This logic is discussed below. This latter applies only for CTPLS and WORKS, of course.

The other basic event is the entry into the shop of a new order. This requires placing that component which has arrived in its queue, after reading in the whole routing of the order. Other components are placed in a "pseudo-queue" to stay until their scheduled arrival time. If a newly released component arrives at a group with an idle machine, provision is made to have it assigned immediately.

There are two other events which are checked continually: one is the reporting period called a "day" (queue length snapshots only) and that called a "period" (at intervals which are restricted to integer days), the other is of course the end of the simulation.

After each event type is processed, and before return to TIMER is made, the next occurrence of that event type is calculated as a candidate, as it were, for the next recursion.

Some Special Programming Features

There are a number of special features of the program that are probably unique in simulation programs if only marginally noteworthy. The order file organization, for instance, involves contiguous (that is, "normal") lists of tasks on segments within threaded lists of segments (there is really no
easier way to express a "tree") constituting the order. Each active segment is "chained" to its machine, or queue, basic date ("task record") in the order file, to its "slack" account, and to its basic order data. There is maintained a "free space" list for the order file. Whenever an order is completed, an index to its space is placed at the top of a "push-down" list. Whenever a new order arrives and takes space, the index to the next vacant spot "pops-up".  

The Rules Themselves

The heart of the program is the SELECT subroutine which contains all of the decision-making in the program. The simpler rules require no special ingenuity in their implementation (simple rules are: FCFSV, SHTOP, TRSIO, EARSD, SLROP). There are some real challenges in the heuristics. Descriptions for GERES, COVERT, CTPLS, and WORKS are shown in Appendix B.

Dynamic Critical Path Analysis

An important computational problem arises when, in simulating the progress of multiple component orders through the shop, it is desired to maintain accurate status information on segment slack and order delay costs. Recall that an order is considered as apt to be tardy as its latest segment and that segments other than the latest (i.e., not on the "critical path") have a relative earliness or slack. Slack represents the delay the segment can tolerate without delaying the order.

---

At any moment in time, every active segment has slack which, unless work is proceeding on its active task, is decreasing. The problem is to maintain timely records on segment slack.

The solution used here goes as follows. Recall that
\[
    c_o = \max_{\alpha \in A} (c) \quad \text{where } c_\alpha \text{ is the tardiness probability of } \alpha's
\]
active operation and
\[
    s_\alpha = (c_o - c_\alpha) \quad \text{is segment slack, } 0 < c_o < 1, \text{ and that}
\]
\[
    T_o = \max_{\alpha \in A} (T_\alpha) \quad \text{where } T_\alpha \text{ is the tardiness of the active operation and}
\]
\[
    s_\alpha = T_o - T_\alpha \quad \text{is segment slack, } c_o = 1.
\]
For "early" orders, \( c_o = 0 \), and segment slack is computed as \( n_i - t_o \), the time until \( c_\alpha \) becomes \( >0 \). These define the figures at any point in time, but what is needed is a way of revising slack and delay cost as time passes.

Suppose that a task is subjected to a known, certain delay of \( t_d \); \( t_d \) would be the time until the next machine in the group becomes free, for example. It can be seen that if \( t_d \leq s_\alpha \), regardless of the level \( c_o \), the new slack becomes \( s_\alpha - t_d \). But if \( t_d > s_\alpha \), a new "critical path" is determined and the order delay cost changes if \( 0 < c_o < 1 \) (neglecting "class changes", i.e., from "early" to "late" or "late" to "tardy", etc.). In particular, the new cost becomes \( c_o = t + t_d - n_i \) where operation \( i \) is the active

\[
    w_i
\]

\[15\] Not for times when a segment is in process. For such cases the lateness or tardiness would be measured from the next scheduled task with a correction for remaining processing time on the current task.
operation in the segment and \( s'_\beta = (c'_o - c_\beta) w_\beta \), \( \beta \notin \alpha \), \( s'_\alpha = 0 \).

The arithmetic for tardy orders is simpler

\[ s'_\beta = s_\beta + t_d - s'_\alpha, \quad s'_\alpha = 0. \]

The principle of the approach is this. When a decision has been made to delay a task for a known time by selecting another task in preference to it, the implications of this delay for all tasks in parallel are promulgated immediately. The general effect is to increase the slack of parallel tasks upon delay of the one or to reduce its own slack for its next turn.

**Experimental Design**

The problem is to design an experimental plan and individual experiments which will provide rigorous tests of the hypotheses, without introducing bias, or costing unduly in computer time--an eminently scarce resource. The problem of arrangement of factors and planning the sequence of experiments is called here "strategic planning", that of determining how the runs themselves are to be executed is called "tactical planning", following Conway. The strategic plan, since it evolved sequentially, is discussed in the next chapter.

**Tactical Plan: Order Generation**

The time between arrivals is obtained by sampling randomly from an exponential (or Erlang) distribution. Other things being equal the load level

\[ ^{16} \text{Op. cit., 47.} \]
is determined by the mean interarrival times, so load adjustment can be made through this parameter. Through use of the transition matrix, order routings may be generated. The matrices used throughout these experiments are those for "rather pure" job shops (nearly equally probable assignment to any other station). The only reason that "perfectly pure" shops are not used is a desire to equalize the level of utilization of the various machine centers while allowing mean processing times to vary. The arrival rates and transition probabilities for various centers must be adjusted to permit this.

The number of operations per order, or per segment in multiple component orders, is set by parameter. Obviously, this could be accomplished by setting up transition probabilities to a "trapping state" as done by Nanot. However, the explicit treatment of this parameter facilitates easy changes in order length and is essentially as efficient computationally. Table 4.1 below contains the particulars for the various types of orders.

**TABLE 4.1**

<table>
<thead>
<tr>
<th>Order type (no. components)</th>
<th>Number of segments</th>
<th>Mean tasks per segment</th>
<th>Range of tasks/segment</th>
<th>Mean tasks per order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (serial)</td>
<td>1</td>
<td>12</td>
<td>8-16</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2-5</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>2.5</td>
<td>1-4</td>
<td>27.5</td>
</tr>
</tbody>
</table>

As each operation is selected by use of the transition matrix, a processing time is selected from the distribution for the particular machine group. These distributions are all "truncated" exponentials with different means. The
truncation occurs at \( t > \mu + 7\sigma \), i.e., at \( \frac{8}{\lambda} \), and does not materially affect the distribution (the frequency of truncation is about 3 per 10,000 tasks) and permits a fixed allocation of storage "bits" for processing time.

To obtain waiting time estimates for purposes of scheduling the orders, a preliminary run is made with the shop using FCFSV. While this might appear to be attributing some sort of omniscience to the simulated shop management, it is really assumed only that the management has the opportunity to simulate the behavior of the shop with its current order backlog. Thus, in real life, more accurate estimates of wait times might be forthcoming; they could in fact, be estimated for the particular rule under study\(^{17}\), and estimated as a varying time series instead of as a time average. The point is that "the deck is not fixed" by this approach assuming reasonably accurate estimates of processing times and access to simulation in real life.

The schedules are developed on the basis of dating each operation on the routing, starting, it may be recalled, from the generated release time, adding the omnibus error term which is selected by sampling randomly from a uniform distribution with center at roughly \(-.3\hat{\sigma}\) of the FCFSV flow distribution and range about \(-.9\hat{\sigma}\) to \(+.3\hat{\sigma}\). That is, the initial "lateness" of the orders will vary between \(-.9\hat{\sigma}\) and \(+.3\hat{\sigma}\). The due date is just the start date of the terminal task plus its processing time.

\(^{17}\) As was done by Rowe, \textit{op. cit.}\n
Summarizing the algebra,

\[ n_1 = r + \varepsilon \]
\[ n_2 = n_1 + t_1 + q_1 \]
\[ \ldots \]
\[ \ldots \]
\[ n_f = n_{t-1} + t_{f-1} + q_{f-1} \]
\[ d = n_f + t_f + q_f \]

for serial orders. For multiple component orders the procedure is more complex. In effect (the algebra is cumbersome), the path having the longest processing plus wait time sum is used to establish the due date and then individual start times are generated by simple backdating.

The release times of the individual initial operations on the paths are set such that they are started "in phase", i.e., all components start with the same initial earliness. In this procedure, the \( u_1 \) for all tasks is calculated in the same way, but without wait time allowances.

The result of these calculations is a large set of orders arranged in release time order. Each file generated thus constitutes the input data for a single set of shop conditions.

The Tactical Problems

Conway has contributed importantly to simulation doctrine with his discussion of the tactical problems of experiments such as this.\(^{18}\) These

\(^{18}\) "Some Tactical Problems...", op. cit.
problems revolve around questions of starting conditions, equilibrium, variability of results (for formal test of hypotheses), sample size, and measurement. He analyzes these problems and proposes doctrine mostly developed from his own experience. He concludes with the following challenge:

"I strongly believe that the manner used in solving these tactical problems has an important influence on the credibility and utility of simulation results. Until such time as there is general agreement on simulation operation, it seems that an investigator reporting his results should have to describe, in considerable detail, how he resolved these issues. Just as the model is described so the reader may judge its suitability, the manner of model operation should be described for the reader's evaluation..." 19

So, on to the resolution of these issues--in considerable detail.

The first problem is that of obtaining measurements from a distribution which approaches the steady state limiting distribution of the system. The first question is how to start up the run since "It takes some time for the simulation to overcome the artificiality introduced by the abrupt beginning of operation..."20 He suggests:

a. exclude data from some initial period from consideration, and

b. choose starting conditions that make the necessary excluded interval as short as possible...

[As to the first] one should make a few pilot runs with very short interval periodic reporting. By plotting this performance against time, one can obtain some idea of what might comprise reasonable deletion.21

---

19 Ibid., 60-61.
20 Ibid., 48.
21 Ibid.
For the second he suggests using one's best a priori guess of mean system state (e.g., queue lengths).

So be it. The procedure is the following. For all runs, use as a starting point the shop conditions "dumped" from the preliminary runs with FCFSV. This is "neutral" system state. That is, there is no serious interest here in FCFSV as an effective discipline. Yet its behavior is neither like the SHTOP rules (as to means), nor the due-date oriented rules such as EARSD and SLROP (as to variance), nor the COVERT family. There is no particular bias then from using a starting point set up in this way and the time to "steady state" should be much shorter than an "empty and idle" start. For purposes of initial exclusion, the suggestion of pilot runs was followed. In fact, even during measurement runs, periodic "snapshots" are taken of queue lengths to test for major biasing transients in the runs. The idea is to run several "non-data" orders through the shop (in practice, 200 were found to be adequate).

The second issue deals with obtaining results which can with credibility be claimed to approximate the limiting state distribution of the system. The challenge is to submit the system to long enough experience such that:

1. Occurrence of "rare events" such as extremely long queues are not overly weighted in the results.
2. A "fair chance" for enough rare events to occur is provided.

More succinctly, one wants a representative sample of system states, and representative means occuring with relative frequency approaching the limiting state distribution.
Instead of attempting to grapple with this dilemma on a theoretical plane, it appears expedient to make the decisions on an empirical basis. The early Cornell research\textsuperscript{22} clearly used runs that were too short (200 orders, perhaps 1000 tasks), Nanot clearly had runs that were more than long enough to estimate the .999 fractile of the flow time distribution, in most cases (up to 114,000 orders, perhaps 500,000 tasks). Since the interest here is fundamentally in the mean value of an attribute (tardiness) of each order, a satisfactory answer undoubtedly lies between (of necessity, closer to the former than the latter). A useful datum is the second set of Cornell experiments in which what appear to be entirely satisfactory results were obtained from 2000 orders, perhaps 10,000 tasks.\textsuperscript{23} Implied, incidentally, in the above, is the opinion that number of tasks is the basic determining factor in the approach of the "temporal" distribution to the limiting distribution.

This history suggests that runs on the order of 20,000 to 30,000 tasks should be adequate. In fact, it was found that 3072 serial orders (about 40,000 tasks in total) and 2048 serial-parallel routed orders (30,000 - 60,000 tasks) were required at utilization levels of .80 - .85.

The second set of problems are those connected with measurement and, more importantly, estimating the precision of the measurement. The measure of effectiveness here and the only measurement which is vital is mean order tardiness. It is clear enough how to measure this attribute for individual orders. It is further quite clear that mean tardiness is the sum of individual

\textsuperscript{22}Conway, Johnson, and Maxwell, "An Experimental...", \textit{op. cit.}

\textsuperscript{23}Conway, and Maxwell, \textit{op. cit.}
measurements dividing by the number of measurements. But the rub is that:

Attributes describing the jobs' experience in the shop and assigned during the course of the simulation are not independent since there is considerable chance variation in conditions over time and contemporary jobs experience very similar conditions.\textsuperscript{24}

That is, the successive observations are autocorrelated. The result of this is a difficulty in estimating the variance of the mean. Conway's recommendation is to

Number the entities [orders] in their order of creation [release to the shop] and divide the sequence into blocks of equal numbers with consecutively numbered entities. For purposes of estimating the precision of results, the block means should be considered basic observations.\textsuperscript{25}

That is, instead of estimating the variance of the mean from

\[ \hat{\sigma}^2_T = \frac{1}{(n-1)} \sum_{1}^{n} (T_j - \bar{T})^2 \]

where \( \bar{T} \) is the "grand" mean tardiness, estimate it by

\[ \hat{\sigma}^2_T = \frac{1}{(m-1)} \sum_{1}^{m} (\bar{T}_j - \bar{T})^2 \]

where the orders are grouped into blocks of size \( \frac{n}{m} \). The successive mean values should be considerably less autocorrelated over time, the larger \( n/m \) the less, of course. The basic block size is 32 (large enough to permit an assumption of normality for the means derived therefrom). However, the autocorrelation in the runs was found to be sufficiently strong that the final block size used was 768. At this size autocorrelation became insignificant.

\textsuperscript{24} Conway, op. cit., 59.

\textsuperscript{25} Ibid., 60.
The last tactical problem, that of finding a procedure for formally testing the hypotheses, is in many ways the most annoying. As Conway points out, the classical analysis of variance model does not apply because it requires assumptions of independent experiments (not true), and common variance (unlikely). He proposes Bechhofer's ranking procedure among other possibilities. 26

In comparing several possible tests for use in this work, including the t-test of paired observations (using large blocks to eliminate autocorrelation), it was found that the "Wilcoxon matched-pairs, signed-rank" test was most powerful. 27 The pairing of (blocked) means corrects for surges in measurements that are characteristic of the input rather than rule performance.

There are two points to be made about these tests. The basic question is whether the results obtained could have arisen by chance. Since the systems are different, having been programmed differently, "failure to reject a null hypothesis only indicates that the test was not sufficiently powerful to detect the difference--e.g., a longer run should have been employed." 28 A more basic point is that unless the results are markedly in favor of the normative hypothesis, visible to the naked eye, so to speak, there is little use in splitting statistical hairs. For the test to have any practical significance, the difference in favor of the proposed rule would have to be substantial.

26 Ibid., 53.
Chapter V
RESULTS

Introduction

In this chapter are discussed the results of the test runs. The strategic plan was this. Using the simplest version of model, that is, single component orders and single channel machine groups, establish the general sensitivity of the model to such factors as level of utilization, the "tightness" of flow allowances, and the "k" parameter used in the COVERT rule to control the approximation to probability of tardiness. Then, based on these results, either to discard factors not found interesting or to "calibrate" subsequent runs so as to capture the main effects. A summary of all experiments, the conclusions for all hypotheses, and an outline of the ex post facto experimental design are all set forth at the end of this chapter.

Sequencing Single Component Orders

The first runs were conducted for the eight machine, single component orders, "pure" job shop, with the utilization level set at .8, and due dates (i.e., flow allowances) established such that about 45 per cent of data orders were late using FCFSV. The basic results are shown in Table 5.1.
TABLE 5.1
Mean Tardiness per Order
Shop 1, Order Type 1, U = .80
3072 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>(\bar{T})</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS</td>
<td>36.6</td>
</tr>
<tr>
<td>EARSD</td>
<td>24.7</td>
</tr>
<tr>
<td>SLROP</td>
<td>16.2</td>
</tr>
<tr>
<td>SHTOP</td>
<td>11.3</td>
</tr>
<tr>
<td>TRSIO</td>
<td>4.6</td>
</tr>
<tr>
<td>COVERT (K=1)</td>
<td>2.5</td>
</tr>
</tbody>
</table>

The time units are, of course, arbitrary, but to provide some basis for comparison, the mean flow time for FCFSV was about 230 time units. The problem of precision of the tardiness measurements is discussed below.

Some insights into the performance of these rules can be gained from the mean wait times achieved. Since all mean processing times were set equal and equiprobable routing was used, each machine, in theory, has the same true mean wait time. Table 5.2 shows the means obtained in these runs. Note that FCFSV, EARSD, and SLROP show no essential difference, to be expected in rules ignoring processing time. The shortest operation rule yields wait times roughly half as large\(^1\), and surprisingly good tardiness results, thus confirming

\(^1\) One can compute the theoretical wait times for FCFSV and SHTOP. For a single work station the ratio of SHTOP to FCFSV wait times is .471, as shown in Conway, "An Experimental Investigation...", op. cit., 102.
The suspicions noted in Chapter III as to its potential efficiency in this type of model.

The rules proposed here (recall that TRSIO is just COVERT with K=0), performed remarkably well; in fact, far better than expected. It is interesting to compare the wait times obtained. TRSIO is nearly as good as SHTOP in this respect; COVERT reflects its nature in trading off tardiness reduction against wait times by falling between the two rules most nearly reflecting its components, namely SHTOP and SLROP.

Some further elucidation of the basic results can be obtained from perusing figure 5.1 which depicts the lateness distributions for these six rules. Lateness, it may be recalled, is completion time less due date. It is quite clear from these histograms that EARSD is not as effective in variance reduction as SLROP. Also, the fundamental difference in shape of distribution as well as mean for SHTOP, relative to the rules which ignore processing time, is highlighted.
FIGURE 5.1: LATENESS DISTRIBUTIONS FOR SIX RULES SHOP 1, ORDER TYPE 1, \( U = .80 \), 3072 Orders
Although the effect is perfectly predictable, the clearly defined bi-modal distribution resulting from the TRSIO rule was interesting to behold. And finally, the compromise nature of COVERT is again clearly illustrated in this figure. Its "right hand tail" is considerably less damaging than that of SHTOP; its mean, considerably lower than SLROP.

At this point, one is already able to draw a limited conclusion. At least under limited circumstances, the COVERT theory, encompassing TRSIO as a special case, is sound and the normative hypothesis is accepted (Hypothesis I, supra, is accepted). It remains to test the generality.

Sensitivity Analysis: Load Effects

It was considered important to test the "load" effects on the results since previous researchers had found some anomalies resulting from this factor. Two additional sets of runs were made, using the same shop configuration and order type, but with utilizations set at .7 and .9 respectively. Table 5.3 presents these results contrasted with those obtained at .8.

These data point up the following phenomena. First, SHTOP is sensitive to load level; note its performance compared to SLROP going upward in utilization. Its long "right hand tail" becomes very long indeed at high load levels and the rule is no longer effective. Second, at a utilization of .7 the differences among rules begin to blur. This suggests that in a lightly loaded shop, sequencing is a less vital activity. But most noteworthy is the continuing superiority of the COVERT family of rules. It should be noted that, in the data presented thus far, only the single parameter setting K=1 has been reported. In no case, it will be seen, was this the best setting of K. Hypothesis III is accepted.
TABLE 5.3

Mean Tardiness per Order
Shop 1, Order Type 1, U = .7, .8, and .9
3072 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\bar{T}$ (.7)</th>
<th>$\bar{T}$ (.8)</th>
<th>$\bar{T}$ (.9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>21.7</td>
<td>36.6</td>
<td>57.9</td>
</tr>
<tr>
<td>EARSD</td>
<td>16.0</td>
<td>24.7</td>
<td>38.6</td>
</tr>
<tr>
<td>SLROP</td>
<td>14.5</td>
<td>16.2</td>
<td>16.8</td>
</tr>
<tr>
<td>SHTOP</td>
<td>7.2a</td>
<td>11.3</td>
<td>28.3</td>
</tr>
<tr>
<td>TRSIO</td>
<td>7.2a</td>
<td>4.6</td>
<td>4.9b</td>
</tr>
<tr>
<td>COVERT (K=1)</td>
<td>4.2</td>
<td>2.5</td>
<td>4.7b</td>
</tr>
</tbody>
</table>

a,b: results not significantly different at .05 level

Sensitivity Analysis: Flow Allowance Effects

In early test runs, a distinct sensitivity of tardiness results to the "tightness" of flow allowances was observed. A full test was then performed, setting "tight allowances", so that about 60 per cent of FCFSV sequenced orders were tardy and loose allowances, so that about 30 per cent were tardy. Table 5.4 shows the results of these tests.

These results illustrate how very sensitive the variance reducing rules are to the due-date setting procedure. As the allowance is tightened, more and more of the "fat" part of the flow distribution falls in the tardy category, so that SLROP, which performs very well (though not so well as COVERT) with generous allowances, is distinctly disadvantageous under "short"

---

2In general one cannot compare absolute mean tardiness figures across columns because flow allowances are set differently. Of interest are changes in relative performance across columns.
TABLE 5.4
Mean Tardiness per Order as Affected by Flow Allowance
Shop 1, Order Type 1, U = .80
3072 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>&quot;Loose&quot; Allowance</th>
<th>&quot;Normal&quot; Allowance</th>
<th>&quot;Tight&quot; Allowance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>13.5</td>
<td>36.6</td>
<td>60.1</td>
</tr>
<tr>
<td>EARSD</td>
<td>4.5</td>
<td>24.7</td>
<td>51.9</td>
</tr>
<tr>
<td>SLROP</td>
<td>1.3a</td>
<td>16.2</td>
<td>50.8</td>
</tr>
<tr>
<td>SHTOP</td>
<td>5.8</td>
<td>11.3</td>
<td>17.0</td>
</tr>
<tr>
<td>TRSIO</td>
<td>1.6a</td>
<td>4.6</td>
<td>20.1</td>
</tr>
<tr>
<td>COVERT (K=1)</td>
<td>0.15</td>
<td>2.5</td>
<td>10.2</td>
</tr>
</tbody>
</table>

a: not significantly different at .05 level

date circumstances, while SHTOP with its spread distribution is less sensitive. This suggest that the flow allowance factor is an extremely important one for studies of this sort and not to be dismissed as cavalierly as it has been in the past.

The results also show that COVERT, but not the naive TRSIO, adapts to whatever circumstance prevails. With loose allowances, it avoids the "long tail" problem; with tight allowances, its shortest operation component tends to dominate. Hypothesis IV is accepted.

Sensitivity Analysis: Setting the COVERT "k"

The setting of K=1, which corresponds to an intercept of slack equal to expected remaining wait time, is quite arbitrary. A series of runs was made to establish the performance characteristics of the particular curve used in the approximation. Table 5.5 reports the results.
TABLE 5.5
Mean Tardiness per Order for COVERT
Shop 1, Order Type 1
3072 Orders

<table>
<thead>
<tr>
<th>K</th>
<th>U = .7</th>
<th>U = .8</th>
<th>U = .9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0(TRSIO)</td>
<td>7.2</td>
<td>4.6</td>
<td>4.9b</td>
</tr>
<tr>
<td>.5</td>
<td>6.4</td>
<td>1.4</td>
<td>3.3</td>
</tr>
<tr>
<td>1</td>
<td>4.2</td>
<td>2.5a</td>
<td>4.7b</td>
</tr>
<tr>
<td>1.5</td>
<td>3.8</td>
<td>2.8a</td>
<td>6.5</td>
</tr>
<tr>
<td>2</td>
<td>5.3</td>
<td>3.4</td>
<td>----</td>
</tr>
</tbody>
</table>

a,b: not significantly different at the .05 level

Aside from the rather pleasing (to the author) regularity of the results, it is clear that COVERT is sensitive to this parameter, and that, in practical application, it would be worthwhile to calibrate the rule. However, since in the experiments, the "reasonable" setting of K=1 provides such superior results relative to all other rules, hereafter, the representative parameter will be used. The following explanation for the particular observed behavior is offered. Consistently, the curve which provides best results is that which ties the curve down to that point just slightly beyond the COVERT expected remaining wait times (as opposed to the FCFSV wait times for which K is the multiplier).

A point should be made about the fact that the wait time estimates used by COVERT (and EARSD, for that matter) are rather crude. Long-run mean values are used and experience indicates rather dramatic departures from the means. For example, in the U = .8 runs for a quarter of the run a mean wait of 15.5 was obtained, compared with the overall mean of 10.3.
Evaluation of Hold-Off and Sneak-In (WORKS)

The heuristics described in Chapter III for modifying COVERT to account for critical, i.e., tardy, arriving orders was subjected to test under all of the conditions described previously. Also, GERES, the heuristic based on SLROP developed by William Gere was tested. Table 5.6 shows the results of these rules juxtaposed with their "base" rules.

**TABLE 5.6**

Mean Tardiness per Order With and Without Hold-Off Heuristics  
Shop 1, Order Type 1  
3072 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>U = .7</th>
<th>U = .8</th>
<th>U = .9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flow Allowances</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Loose</td>
<td>Normal</td>
<td>Tight</td>
</tr>
<tr>
<td>SLROP</td>
<td>14.5a</td>
<td>1.3</td>
<td>16.2</td>
</tr>
<tr>
<td>GERES</td>
<td>14.5a</td>
<td>1.0</td>
<td>19.2</td>
</tr>
<tr>
<td>COVERT (.5)</td>
<td>-----</td>
<td>-----</td>
<td>1.4d</td>
</tr>
<tr>
<td>WORKS (.5)</td>
<td>-----</td>
<td>-----</td>
<td>1.3d</td>
</tr>
<tr>
<td>COVERT (1)</td>
<td>4.2b</td>
<td>0.15c</td>
<td>2.5</td>
</tr>
<tr>
<td>WORKS (1)</td>
<td>4.0b</td>
<td>0.20c</td>
<td>2.0</td>
</tr>
<tr>
<td>COVERT (1.5)</td>
<td>3.8b</td>
<td>-----</td>
<td>3.4</td>
</tr>
<tr>
<td>WORKS (1.5)</td>
<td>3.8b</td>
<td>-----</td>
<td>2.7</td>
</tr>
</tbody>
</table>

a,b,c,d: not significantly different at the .05 level

One can conclude from these data first of all that the relatively tight flow allowances generally used here bias the results of Gere's heuristics;

---

for example, note the perfectly respectable performance obtained with the loose allowance. Since his heuristics apparently do not substantially affect mean flow times, rather expending their energy in further variance reduction, one must conclude that the heuristics must be useful under conditions in which SLROP is generally good, namely with generous allowances.

The hold-off modification to COVERT appears to be marginally effective. Neglecting the loose allowance case (due to its essentially unmeasurable tardiness), WORKS does no worse than COVERT, and in several cases provides statistically significant, if economically modest, improvement. There appears to be some evidence that the improvement is greater at higher load levels. The queue length cut-off (the queue size at which hold-off was prohibited) was run throughout at n=6, a general insensitivity to this parameter having been inferred from test run results. Hypothesis II is accepted.

The Measurement Problem

The procedure of calculating mean tardiness from blocks of 64 successively released orders and computing signed ranks for paired blocks provides a way of comparing results (for pairs of rules) from these tests. But a glance at figure 5.2 should illustrate why no "precision" measurement accompanies the means as reported in the text. First, note the considerable autocorrelation present even after blocking 128 orders. But, much more important, note that the series for FCFSV, SHTOP, and COVERT (.5) never intersect. There is perfect dominance in this case. Yet, the means and estimated standard deviations for the three rules are: FCFSV, \( \hat{\mu} = 36.6, \hat{\sigma} = 8.6 \), SHTOP, \( \hat{\mu} = 11.3, \hat{\sigma} = 4.0 \); and COVERT, \( \hat{\mu} = 1.4, \hat{\sigma} = .9 \); (all based on block sizes of 768 orders), which figures imply considerable overlap in the distributions.
FIGURE 5.2: MEAN TARDINESS OF 24 SUCCESSIVE BLOCKS OF 128 SUCCESSIVELY RELEASED ORDERS (U = .80)
The estimated standard deviations are reported in Appendix D.

**Some Limited Conclusions**

Having tested the sensitivity to some of the more obvious factors, it seems within the realm of prudence to conclude that the COVERT rule is in fact generally efficacious for the simplest model and, moreover to accept the hypothesis that hold-off and sneak-in are effective, if only marginally, adjuncts to the basic rule. It now remains to complicate the model.

**Single Component Orders with Multiple Channel Machine Groups**

Two specialized shop configurations were set up to test the effects of having more than one machine per group: shop 2 has eight groups, each having two identical machines; shop 3 has eight groups, each having four identical machines. The set of rules was modified by dropping EARSD, due to general ineffectiveness, Gere, because of model differences, and adding CTPLS, a COVERT-type rule based on \( \tau \) instead of \( t \). (viz. Chapter III). Also WORKS was modified to include \( \tau \). These two rules have a parameter, \( v \), which governs the substitution of \( \xi \) (the time until the next machine becomes free) for \( t \). Two other minor modifications were tested as well. One changed the tie-break rule in COVERT to earliest urgent start date instead of processing time. Call this COTUTB. The other is the "additive rule" which adds \( (m-1)\mu_t \) to the denominator of COVERT as explained in Chapter III. Call this rule COTADD.
TABLE 5.7

Mean Tardiness per Order
Order Type 1
2048 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>Shop 2 (U = .83)</th>
<th>Shop 3 (U = .85)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>28.7</td>
<td>28.2</td>
</tr>
<tr>
<td>SLROP</td>
<td>18.4</td>
<td>17.9</td>
</tr>
<tr>
<td>SHTOP</td>
<td>7.9</td>
<td>9.8</td>
</tr>
<tr>
<td>TRSIO</td>
<td>5.7</td>
<td>4.2</td>
</tr>
<tr>
<td>COVERT (v=0)</td>
<td>3.1a</td>
<td>2.0d</td>
</tr>
<tr>
<td>(v=2)</td>
<td>3.4b</td>
<td>2.3d</td>
</tr>
<tr>
<td>(v=4)</td>
<td>4.0</td>
<td>-----</td>
</tr>
<tr>
<td>COTUTB (v=0)</td>
<td>3.3b</td>
<td>2.1d</td>
</tr>
<tr>
<td>(v=2)</td>
<td>3.2a,b</td>
<td>2.2d</td>
</tr>
<tr>
<td>(v=4)</td>
<td>4.4c</td>
<td>-----</td>
</tr>
<tr>
<td>WORKS (v=0)</td>
<td>2.9a</td>
<td>2.0d</td>
</tr>
<tr>
<td>(v=2)</td>
<td>3.1a</td>
<td>2.3d</td>
</tr>
<tr>
<td>(v=4)</td>
<td>3.7</td>
<td>-----</td>
</tr>
<tr>
<td>COTADD</td>
<td>4.4c</td>
<td>3.4</td>
</tr>
</tbody>
</table>

a,b,c,d: not significantly different at .05 level

From the results shown in table 5.7, the following inferences seem defensible.

First, the "T" modification is ineffective. Despite the fact that no significant differences were obtained with v=2, the essential (non-parametric) fact is that in no case did it do better. Second, COTADD is not interesting. Third, the general conclusions drawn for the simple model appear to extend to the multiple channel case; that is, COVERT is highly effective and the hold-off, sneak-in heuristic provides, at most, marginal improvement. That is, Hypotheses V and VI are rejected, and VII is accepted.
The General Shop

A set of runs were made with a shop consisting of five single machine groups, two two machine groups, and one four machine group in order to test for possible interactions. The utilization varied from .75 to .85 for the single channel groups, .80 to .90 for the two channel groups, and was set at .86 for the four channel group. The overall shop utilization was about .84. These results which were obtained are shown in table 5.8.

**TABLE 5.8**

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\bar{T}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>24.6</td>
</tr>
<tr>
<td>EARSD</td>
<td>14.1</td>
</tr>
<tr>
<td>SLROP</td>
<td>8.6</td>
</tr>
<tr>
<td>SHTOP</td>
<td>8.0</td>
</tr>
<tr>
<td>TRSIO</td>
<td>3.0</td>
</tr>
<tr>
<td>COVERT (K=1, v=0)</td>
<td>2.8</td>
</tr>
<tr>
<td>WORKS (K=1, v=0)</td>
<td>2.6</td>
</tr>
</tbody>
</table>

While no surprises were lurking in these runs, some value was derived from the replication. Once again, the general pattern of behavior was confirmed.

In general then, the conclusion is drawn that COVERT and WORKS are effective rules for the given models under rather general conditions, for single component orders. Formally, Hypothesis VIII is accepted.
Sequencing Multiple-Component Orders

Some changes were introduced into the procedure to facilitate the testing of rules for sequencing multiple component orders. First, the distribution of interarrival times was changed to Erlang-3 from exponential. Test runs indicated a problem in equilibrium from introducing such large collections of work content with the exponential. This begs the validity question. There seems no compelling empirical reason to choose exponential arrivals over Erlang; the choice appears to have been dictated by the desire to maintain close relationship with analytical work. Erlang arrivals do "settle the shop down" to a considerable degree. The second change was to institute the measuring of "staging time", the time one component waits for another before joining the queue at a station involving their assembly. As previously discussed, this figure has relevance in determining the flow times of individual orders.

Three sets of runs were run with multiple component orders. The first set returns to the simple shop configuration, but uses as inputs three component orders. The second runs were based on shop 2, the two-machine-per-group configuration and the same order type. Replication with "tight" due dates was made here. The final set of runs once again used shop 1, but with six component orders.

The results from the three components, shop 1 runs are tabulated in table 5.9.

Note in that table first, the marked superiority of SLROP over SHTOP. As conjectured in Chapter III, the high spread of the SHTOP distribution causes a price to be paid in staging time by this rule. SLROP's low variance
TABLE 5.9
Mean Tardiness per Order
Shop 1, Order Type 2, U = .8
2048 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\bar{T}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>40.5</td>
</tr>
<tr>
<td>EARSD</td>
<td>17.2</td>
</tr>
<tr>
<td>SLROP</td>
<td>5.9</td>
</tr>
<tr>
<td>SHTOP</td>
<td>16.2</td>
</tr>
<tr>
<td>TRSIO</td>
<td>6.3</td>
</tr>
<tr>
<td>COVERT (K=1)</td>
<td>1.9</td>
</tr>
<tr>
<td>CTPLS (K=1)</td>
<td>2.3</td>
</tr>
<tr>
<td>WORKS (K=1)</td>
<td>1.4</td>
</tr>
</tbody>
</table>

reduces the "out-of-phaseness". Waiting and staging time statistics for this run are shown in table 5.10.

TABLE 5.10
Mean Wait and Staging Times
Shop 1, Order Type 2, U = .8
2048 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\bar{W}$</th>
<th>$\bar{S}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>12.7</td>
<td>28.5</td>
</tr>
<tr>
<td>EARSD</td>
<td>12.6</td>
<td>20.5</td>
</tr>
<tr>
<td>SLROP</td>
<td>12.5</td>
<td>24.0</td>
</tr>
<tr>
<td>SHTOP</td>
<td>5.9</td>
<td>29.4</td>
</tr>
<tr>
<td>TRSIO</td>
<td>6.4</td>
<td>30.4</td>
</tr>
<tr>
<td>COVERT</td>
<td>9.1</td>
<td>23.8</td>
</tr>
<tr>
<td>CTPLS</td>
<td>9.3</td>
<td>21.2</td>
</tr>
<tr>
<td>WORKS</td>
<td>9.3</td>
<td>21.4</td>
</tr>
</tbody>
</table>
These data confirm the presence of this mechanism. The TRSIO rule loses some of its effectiveness in this more complicated situation as well. As can be seen, its staging times are the highest of all.

COVERT continues to show a persistent robustness. In spite of its simplicity, it performs better than the highly complex CTPLS, the latter's access to elegantly maintained "critical path" information apparently notwithstanding. However, the same information forms the basis for WORKS, which includes a hold-off, sneak-in heuristic, and this rule does perform somewhat better in this situation. But, it is clear already that no breakthrough beyond the COVERT barrier has taken place. Hypothesis IX is therefore rejected.

The next runs used the two channel-per-machine-group configuration. The purpose was again to generalize the results to different configurations and, as before, to test special heuristics for the multiple channel case.

The heuristic in this situation calls for seeking an alternate machine for the highest ranked order, the general theory being that in this way, a zero delay cost assignment could be thus preplanned.

The results of these runs, with "normal" and "tight" flow allowances are shown in table 5.11. The pattern repeats as to essential findings, although CTPLS does show up as superior with stringent allowances. Once again, the persistent moderate improvement from the hold-off rules obtained.

The alternate machine heuristic showed no promise, and Hypothesis X is rejected.
TABLE 5.11

Mean Tardiness per Order
Shop 2, Order Type 2, U = .83
2048 Orders

<table>
<thead>
<tr>
<th>Rule</th>
<th>Normal Allowance</th>
<th>Tight Allowance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFSV</td>
<td>27.1</td>
<td>47.7</td>
</tr>
<tr>
<td>EARSD</td>
<td>12.4</td>
<td>24.1</td>
</tr>
<tr>
<td>SLROP</td>
<td>4.8</td>
<td>17.2</td>
</tr>
<tr>
<td>SHTOP</td>
<td>12.3</td>
<td>20.8</td>
</tr>
<tr>
<td>TRSIO</td>
<td>4.4</td>
<td>19.4</td>
</tr>
<tr>
<td>COVERT</td>
<td>2.7a</td>
<td>13.8</td>
</tr>
<tr>
<td>CTPLS</td>
<td>2.7a</td>
<td>12.3b</td>
</tr>
<tr>
<td>CTPLS (alt. machine)</td>
<td>3.2</td>
<td>12.9</td>
</tr>
<tr>
<td>WORKS</td>
<td>2.6a</td>
<td>11.9b</td>
</tr>
<tr>
<td>WORKS (alt. machine)</td>
<td>2.9</td>
<td>----</td>
</tr>
</tbody>
</table>

a,b: not significantly different at the .05 level

The final set of test runs were made using shop 1, but with six component orders, in order to explore the effects of the order complexity dimension. The results are presented in table 5.12.

These data tend to show the previously observed phenomena more clearly. SHTOP, for instance, does exceedingly poorly; SLROP very well indeed. COVERT once again demonstrates a clear superiority. And all hope for a significant improvement through the elaborate heuristic processes of CTPLS and WORKS is dashed. Hypotheses XI and XII are rejected. Throughout the multiple component runs, the hold-off heuristic has performed as before, as well or better than COVERT, so Hypothesis XIII is accepted.
Intuitively, it would seem that the problem is with the heuristic, or perhaps the test conditions, and not that the vast "data base", the critical path information, is valueless. But it is abundantly clear that the value of this information process cum heuristics making use of it has not been demonstrated in these experiments.

The hypotheses and conclusions from test runs are shown in table 5.13; the *ex post facto* experimental design is depicted in table 5.14.
### TABLE 5.13

Summary of Hypotheses and Conclusions

<table>
<thead>
<tr>
<th>No.</th>
<th>Hypothesis Statement</th>
<th>Conclusion</th>
<th>Reference to Supporting Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>COVERT effective for shop 1, order set 1, (serial), ( U = .8 )</td>
<td>Accept</td>
<td>Table 5.1</td>
</tr>
<tr>
<td>II</td>
<td>Hold-off, sneak-in effective for serial orders, shop 1</td>
<td>Accept</td>
<td>Table 5.6</td>
</tr>
<tr>
<td>III</td>
<td>I and II hold regardless of level of ( U )</td>
<td>Accept</td>
<td>Table 5.3</td>
</tr>
<tr>
<td>IV</td>
<td>I and II hold regardless of &quot;tightness&quot; of flow allowance</td>
<td>Accept</td>
<td>Table 5.4</td>
</tr>
<tr>
<td>V</td>
<td>Multiple channel modification of COVERT is effective</td>
<td>Reject</td>
<td>Table 5.7</td>
</tr>
<tr>
<td>VI</td>
<td>The &quot;additive rule&quot;, COTADD, is effective for multiple channels</td>
<td>Reject</td>
<td>Table 5.7</td>
</tr>
<tr>
<td>VII</td>
<td>Hold-off is effective for multiple channels</td>
<td>Accept</td>
<td>Table 5.7</td>
</tr>
<tr>
<td>VIII</td>
<td>COVERT plus hold-off is effective independent of shop channel configuration</td>
<td>Accept</td>
<td>Table 5.8</td>
</tr>
<tr>
<td>IX</td>
<td>The multiple component modification of COVERT is effective for three-component orders (CTPLS)</td>
<td>Reject</td>
<td>Table 5.9</td>
</tr>
<tr>
<td>X</td>
<td>The multiple channel modification for multiple component orders is effective</td>
<td>Reject</td>
<td>Table 5.11</td>
</tr>
<tr>
<td>XI</td>
<td>CTPLS is effective for six component orders</td>
<td>Reject</td>
<td>Table 5.12</td>
</tr>
<tr>
<td>XII</td>
<td>CTPLS is relatively more effective for 6- than 3-component orders</td>
<td>Reject</td>
<td>Tables 5.9, 5.12</td>
</tr>
<tr>
<td>XIII</td>
<td>Hold-off is effective for multiple component orders</td>
<td>Accept</td>
<td>Tables 5.9, 5.11, 5.12</td>
</tr>
</tbody>
</table>
TABLE 5.14
Retrospective Experimental Design

<table>
<thead>
<tr>
<th>Rule</th>
<th>FCFVS</th>
<th>EARSD</th>
<th>SLROP</th>
<th>SHTOP</th>
<th>GERES</th>
<th>TRSIO</th>
<th>COVERT*</th>
<th>CTPLS</th>
<th>WORKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0_1 s_1 L_1 \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_1 s_1 L_2 \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_1 s_1 L_2 \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_1 s_1 L_3 \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_1 s_2 \tilde{L} \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_1 s_3 \tilde{L} \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_1 \tilde{S} \tilde{L} \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_2 s_1 L_2 \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_2 s_2 \tilde{L} \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$0_3 s_1 L_2 \tilde{D}$</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

* several parameter levels tested

$0_1$ serial orders
$0_2$ three-component orders
$0_3$ six-component orders
$S_1$ single channel machine groups
$S_2$ two-channel machine groups
$S_3$ four-channel machine groups
$\tilde{S}$ mixed configuration

$L_1$ $U = .7$
$L_2$ $U = .8$
$L_3$ $U = .9$
$\tilde{L}$ mixed utilization
$D_1$ "loose" due dates
$\tilde{D}$ "normal" due dates
$D_2$ "tight" due dates
Chapter VI
SUMMARY AND CONCLUSIONS

Introduction

The problem of sequencing in job shops was undertaken because of its nature as a generic large, sloppy, system problem. Consequently, the conclusions one can draw from the theory and experiments can be extended to a variety of situations and interpreted at several levels. Of primary interest here is the basic problem of job shop dispatching, in its narrowest interpretation. But one would be remiss not to spell out findings with regard to heuristic programming, on-line real-time systems value, and the lessons learned about experimental method.

In this chapter, these conclusions are drawn following a brief summary of goals, procedure and results. In turn, these are followed by a brief description of collateral research and suggestions for further study.

Summary of Research

The purpose of this study was to find a better sequencing system and demonstrate its effectiveness using models which incorporate some of the complexities found in real life. In particular, two major extensions of the classical model were made, those of multiple channel machine groups and multiple component orders.

A new family of sequencing rules, called collectively COVERT, was developed to provide a basis for economically rational heuristics to cope with these complications.
A new criterion and set of assumptions was proposed for this work. By considering due dates and release times as exogenously supplied (as others have) and by assuming a prohibition on early shipments, the only variable costs are incurred when orders become tardy. It was further assumed that these costs were linear and that all orders incurred penalties at the same rate, thus providing a single measure of effectiveness for the comparison, mean (or, equally satisfactory, total) order tardiness.

A simulation model was constructed to test these rules against several which had been previously proposed, and a semi-factorial set of experiments was conducted.

The proposed family of rules was found to be highly effective under all of the circumstances tested. But, time and again, the simplest rule, the unmodified COVERT, performed nearly as well as, and in some instances better than, the elaborate heuristics designed to improve it. As noted earlier, this precisely reverses Gere's findings as to the relative merits of basic priority mechanisms and heuristics which modify them.

Implications for Job Shop Sequencing

The basic objective of this work has been met, namely, to find operationally useful systems for sequencing in circumstances more nearly approaching those of real life. The basic efficacy of the COVERT rule would appear to be indubitable, its robustness having been demonstrated by superior performance under all conditions tested and further attested to by the generally good performance of its most naive version, TRSIO.

Moreover, the extension of the rule to consider arriving orders for
purposes of analyzing "hold-off" and "sneak-in", has been found to be rather consistently effective, although the improvement is not of an order to be exciting.

A number of test runs were made with the rules, fundamentally for the purpose of insuring the proper operation of the computer programs. But, in that extreme circumstances were generally used for the tests (to probe low probability "branches" of the program), in total, their outcomes shed some further insight. In every condition tested, the COVERT rule and modifications, were dominantly superior.

The COVERT rule without its modifications is a very simple rule to use. In practice, one could calculate a constant (prior to release)

\[ M = \frac{1}{(u_i - n_i) \cdot t_i} \]

where

\[ u_i = \text{"urgent start date" } = d - \sum_{j=i+1}^{n} t_j, \]

\[ n_i = \text{"normal start date" } = d - \sum_{j=i+1}^{n} t_j - \sum_{j=i}^{n} q_j, \]

\[ t_i = \text{processing time of the imminent operation.} \]

These items of information are required by other rules such as EARSD, SLROP, or SHTOP. Then, upon dispatching at time \( t \), one would have only to calculate

\[ L = t - n_i \]

the order lateness. The priority index would be

\[ \pi = L \cdot M \]

and the task having the highest \( \pi \) would be selected. This is not to beg the
question of machinists' mathematics; it is meant to show that COVERT is no more difficult to implement, in principle, than SLROP, for example.

The efficacy of COVERT apparently derives from its ability, despite the crudeness of its approximation, to effect a trade-off between potential tardiness (or generically, delay costs) and processing time for the tasks. It preserves the good features of its component rules, SLROP and SHTOP, and eliminates the bad features, considered at the physical level. Furthermore, it is trivial to add real-life complexities such as non-linear tardiness penalties, inventory carrying costs, and differing penalties among orders to COVERT. Yet, since none of the other rules have an economic underpinning, they cannot respond to such circumstances. Parenthetically, it should be pointed out that the incorporation of such features into the model was avoided for reasons of experimental conservatism. That is, I wanted to avoid setting up "straw men" as competitors to COVERT, the more flexible rule.

It is my intuition that the multiple component modification of COVERT can be improved to the point at which it is significantly superior to the basic rule, statistically and economically. The modification as it exists is quite complex (over 1000 FAP instructions for the rule and critical path updating routines). There are numerous possible parameter values to test and features to modify. This feeling echoes that voiced by Conway in his RAND work "...the writer is convinced that a rule of the form 917 P+XWQ... can be shown to be superior to SHOPN [SHTOP]..."\(^1\) This will be pursued. From the results obtained here, the sensible conclusion for the practitioner would be to use COVERT in its simplest form.

\(^1\)"An Experimental Investigation...", op. cit., 63.
This statement should be qualified, however. COVERT has been shown to be effective only within the class of real-time sequencing rules. It remains to ascertain the relative merits of real-time rules versus preplanned sequences, such as might be obtained from integer programming (assuming the computational limitations of that approach are alleviated).

Implications for Heuristic Programming

There was one major lesson learned in this work about heuristics. This was that for rules to be used in a dynamic environment, static analysis and test can be misleading. Now, Forrester has clearly pointed out the limits of intuition in understanding dynamic systems, but this work provides one more piece of evidence. For example, the elaborate heuristics to modify COVERT for the multiple component case showed great promise in hand simulation, but obviously this was not valid. In observing the time series of order tardiness, it became clear that the heuristic breaks down under conditions of dynamically increasing load.

The form of heuristic employed in these experiments is of a simple type, structurally. Prune the combinational tree based on a ranking procedure and search exhaustively the remaining branches. This approach was shown to have merit in chess programs by Newell et al., and is computationally attractive. It seemed the obvious type to use because of the presence of a demonstrably good ranking procedure.

---


One wonders, however, if other types of heuristics might be found, perhaps by observing the humans who perform the dispatching function. This would be a valuable study. But the problem of human incapacity in the face of complex dynamic phenomena would remain.

On-Line, Real-Time Systems

The conclusions drawn about the value of the extended current-data base would have to be similar to those drawn by Conway earlier. That is, no substantial value has been shown. The burden remains on the advocates of this type of system (and I admit to being one) to demonstrate its value. COVERT can be applied locally, based on precalculated information. The modifications (CTPLS and WORKS) would require access to the current, central file. The fact that these modifications showed at most modest improvement leaves the value question unanswered.

It should certainly be pointed out, however, that research being performed on this same model by Francis Russo, in investigation of the "alternate routing" problem, tends to show type of substantial value that would justify these systems.4 His more elaborate heuristics for selecting which of a partially ordered set of operations to perform next on an order are massively effective and require OLRT implementation.

The Model and Experimental Method

Considerable merit is claimed for the economic framework of the model. 4

---

It includes what is felt to be a reasonable emphasis on the tardiness problem and the assumption of no early shipments seems at least as viable as complete freedom in this respect. But its greatest charm is the reduction of all performance measurement to a single measure of effectiveness. No equivocation on relative performance is needed.

Use of a fixed level of utilization model also seems reasonable in retrospect. It seems unfair somehow (as in the fixed backlog model) to expect the sequencing system to "create" order inputs, this function being at least partially the problem of the sales force, and other exogenous (to the production organization) agencies. Conway's switch to this type of model from the fixed backlog type seems to confirm this judgement.

All of the "tactical problems" predicted by Conway were present in this study. His proposed solutions worked and his discussion provided an invaluable guide.

One aspect of simulation experiments as contrasted to biological treatments, say, deserves emphasis. This is the general ability to simulate simultaneity of conditions by using identical inputs while experimenting sequentially. The monday morning quarterback approach becomes possible. One can perform experiments on ideas generated during the course of an experiment yet give the appearance of (and truly simulate) simultaneous experimentation. The upshot of this faculty is that one need not commit himself to a fully factorial design of experiments. He need only carry along factors which prove important in early experiments.

Collateral Research and Further Study

The Russo work based on the model developed here has been previously
cited. It is concerned with the "alternate routing" problem defined in Chapter II. A similar series of experiments are being performed by Paul Clermont on the related but different problem denoted as "machine substitution", that is, the case in which one machine may draw work from another under certain (non-reversible) circumstances. 5 Also in parallel with this work has been performed an exploratory study of feedback phenomena in heavily loaded shops by Lalit Kanodia. 6

All of these studies are aimed at determining further strategies for low-level decision-making given additional controllable variables to consider. The evidence so far (certain in the case of the cited theses) is that, to the degree that freedom of choice in routing, in machine selection, or in processing rates obtains, taking advantage of this freedom may be more important than the sequencing discipline used.

Research on this subject area will continue along several avenues. One is to study the improvement of the COVERT probability estimation, another is to seek improvements of the multiple component heuristics. But the main thrust will be towards understanding the next higher level decision process called "structuring" in Chapter II. Experiments are projected for studying man-machine cooperation on shop parameter setting. The work reported in this thesis provides a base from which to move upwards in the decision hierarchy.

---

5 The title of this thesis has not been selected as of this writing.

APPENDIX A

The Influence of Mean and Variance on Normal Quadratic Tardiness Loss Functions

Assume flow times, x, are normally distributed with mean µ and standard deviation σ. Assume that allowable flow time, a, is constant for all orders. The tardiness loss function is quadratic,

\[ L(x - a) = c_1 \int_{a}^{x} (x - a)^2 f(x) \, dx + c_2 \int_{a}^{x} (x - a) f(x) \, dx + c_3 \int_{a}^{x} f(x) \, dx \]  

(1)

where

\[ f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2} . \]

Let \( \phi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \), \( \Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{0}^{u} e^{-t^2/2} \, dt \), and \( u = \frac{a - \mu}{\sigma} \).

Plan

It is proposed to obtain expressions for each term of (1) as a function of \( u \), \( \phi(u) \), \( \Phi(u) \), and \( \sigma \), and to obtain derivatives of each term with respect to \( \mu \) and \( \sigma \), (2) - (10). These are combined to obtain \( \frac{\partial L}{\partial \mu} \) and \( \frac{\partial L}{\partial \sigma} \) in (12) and (13). The derived expression for \( L \) is (11).

1. Constant term:

\[ k_3 = c_3 \int_{a}^{x} f(x) \, dx = c_3 \phi(u) \]  

(2)

\[ \frac{\partial k_3}{\partial \mu} = c_3 \cdot \frac{\partial \phi(u)}{\partial u} \cdot \frac{\partial u}{\partial \mu} = c_3 \cdot -\phi(u) \cdot \frac{-1}{\sigma} = c_3 \cdot \frac{\phi(u)}{\sigma} \]  

(3)
\[ \frac{\partial k_3}{\partial \sigma} = c_3 \frac{\partial (u \phi(u))}{\partial u} - \frac{\partial u}{\partial \sigma} = c_3 \cdot -\phi(u) \cdot \frac{(a-\mu)}{\sigma^2} \]

\[ = c_3 \cdot \frac{\partial \phi(u)}{\partial u} \]

Note that \[ \frac{\partial k_3}{\partial \sigma} > \frac{\partial k_3}{\partial \mu} \]
whenever \( u \gtrsim 1 \), i.e., \( a - \mu \gtrsim \sigma \).

2. Linear term:

\[ K_2 = c_2 \int_a^\infty (x-a) f(x) \, dx = c_2 \left[ \int_a^\infty \frac{x}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}(x-u)^2} \, dx - a \Phi(u) \right]. \]

To perform the integration,

let \( t = \frac{x-u}{\sigma} \), giving

\[ \frac{1}{\sqrt{2\pi}} \int_u^\infty t \, e^{-t^2/2} \, dt = \frac{-t}{\sqrt{2\pi}} \int_u^\infty e^{-t^2/2} \, dt + \Phi(u) \]

\[ = \frac{-t}{\sqrt{2\pi}} \left[ \frac{-t^2/2}{u} \right] + \frac{\Phi(u)}{\Phi(u)} = \Phi(u) + \Phi(u). \]

Substituting in the full expression,

\[ k_2 = c_2 \left[ \sigma \phi(u) + (\mu-a) \Phi(u) \right] \]

\[ = c_2 \sigma [\phi(u) - u \Phi(u)]; \]

\[ \frac{\partial k_2}{\partial \mu} = c_2 \sigma [\phi'(u) \frac{\partial}{\partial \mu} \frac{u \phi(u)}{\sigma} + \Phi(u)] \]

\[ = c_2 \sigma [-\phi'(u) \frac{\partial}{\partial \mu} \frac{u \phi(u)}{\sigma} + \Phi(u)] \]

Now, note \( u \phi(u) = -\phi'(u) \), so

\[ \frac{\partial k_2}{\partial \mu} = c_2 \Phi(u); \]

\[ \frac{\partial k_2}{\partial \sigma} = c_2 \phi(u) \] by similar algebra.

Note that \( \frac{\partial k_2}{\partial \sigma} > \frac{\partial k_2}{\partial \mu} \)
whenever \( u \gtrsim 0.3 \) (from a table of normal probabilities).
3. Squared term:

\[ k_1 = c_1 \int_a^\infty (x-a)^2 f(x) \, dx \]
\[ = c_1 \left[ \int_a^\infty x^2 f(x) \, dx - 2a \int_a^\infty xf(x) \, dx + a^2 \int_a^\infty f(x) \, dx \right] \]

Letting \( t = \frac{x-\mu}{\sigma} \) again,

\[ k_1 = c_1 \left[ \int u (\sigma^2 t + \mu)^2 \phi(t) \, dt - 2a \sigma \phi(u) - 2a \mu \Phi(u) + a^2 \Phi(u) \right] . \]

Performing the integration

\[ \sigma^2 \int u t^2 \phi(t) \, dt + 2 \sigma \mu \int u t \phi(t) \, dt + \mu^2 \Phi(u) \]
\[ = \sigma^2 \int u \left( \frac{-t}{2\pi} \right) (-t e^{-t^2/2}) \, dt + 2 \sigma \mu \phi(u) + \mu^2 \Phi(u) . \]

Integrating by parts and applying L'Hôpital's Rule,

\[ = \sigma^2 u \phi(u) + \sigma^2 \Phi(u) + 2 \sigma \mu \phi(u) + \mu^2 \Phi(u) . \]

Substituting into the full expression,

\[ k_1 = c_1 [ (\sigma^2 u + 2 \sigma \mu - 2 \sigma \phi(u)) \phi(u) + (\sigma^2 \mu^2 - 2 a \mu + a^2) \Phi(u) ] \]

Now

\[ -2 \sigma (a-\mu) = -2 \sigma^2 u \quad \text{and} \quad \mu^2 - 2 a \mu + a^2 = \sigma^2 u^2 \]

so

\[ k_1 = c_1 \sigma^2 [ (u^2 + 1) \Phi(u) - u \phi(u) ] \quad (8) \]

\[ \frac{\partial k_1}{\partial \mu} = c_1 \sigma^2 [ (u^2 + 1) \frac{\phi(u)}{\sigma} - \Phi(u) \cdot \frac{2u}{\sigma} + \frac{u \phi'(u)}{\sigma} + \frac{\phi(u)}{\sigma} ] \]
\[ = c_1 \sigma^2 [ u^2 \frac{\phi(u)}{\sigma} - 2u \Phi(u) - u^2 \frac{\phi(u)}{\sigma} + \frac{\phi(u)}{\sigma} ] \]
\[ = 2c_1 \sigma [ \phi(u) - u \Phi(u) ] \quad (9) \]
\[ \frac{\partial^k_1}{\partial \sigma} = c_1 \sigma^2 \left[ (u^2 + 1) \frac{u \phi(u)}{\sigma} - \Phi(u) \right] \frac{2u^2}{\sigma} + \frac{u^2 \phi'(u)}{\sigma} + \frac{\phi'(u)u}{\sigma} + 2c_1 \sigma \left[ (u^2 + 1) \Phi(u) - u \phi(u) \right] \]
\[ = c_1 \left[ u^3 \phi(u) + u \phi(u) - 2u^2 \Phi(u) - u^3 \phi(u) + u \phi(u) \right] + 2u^2 \Phi(u) + 2 \Phi(u) - u \phi(u) \]
\[ = 2c_1 \sigma \Phi(u). \quad (10) \]

Note that
\[ \frac{\partial^k_1}{\partial \sigma} > \frac{\partial^k_1}{\partial \mu} \quad \text{whenever} \quad u > -5. \]

Summing over all terms
\[ \mathcal{L} = c_1 \sigma^2 \left[ (u^2 + 1) \Phi(u) - u \phi(u) \right] + c_2 \sigma \left[ \phi(u) - u \Phi(u) \right] + c_3 \Phi(u) \quad (11) \]
\[ \frac{\partial \mathcal{L}}{\partial \mu} = 2c_1 \sigma \left[ \phi(u) - u \Phi(u) \right] + c_2 \Phi(u) + c_3 \frac{\phi'(u)}{\sigma}, \quad (12) \]
\[ \frac{\partial \mathcal{L}}{\partial \sigma} = 2c_1 \sigma \Phi(u) + c_2 \phi(u) + c_3 \frac{u \phi(u)}{\sigma}. \quad (13) \]
APPENDIX B

Description of Selected Heuristics

1. **GERES**

   The flow chart for GERES is shown in Figure B.1. The procedure is as follows: First all tasks are ranked by minimum slack per remaining operation. Arriving tasks are ranked separately. The ranking arrives at the selection of one present task (Gere's "standby job"), task A.

   The next phase of the heuristic is "alternate operation". First, the time of completion of the standby job is computed; then all other tasks are ranked as if the standby job had been selected. This ranking results in a second choice ("alternate standby job"), task B. Then, A's priority index, computed assuming that B is selected, is compared with B's, computed assuming that A is selected. If A's is less or equal, it is selected; otherwise B is.

   The heuristic then enters its "look ahead" phase. If the selected job is critical, or if there is no arriving job, no analysis is made. The highest priority arriving job is tested to see whether, if it is selected, it makes the previously selected task tardy. If so, no hold-off is made; if not, a hold-off is indicated.

   "Sneak-in" analysis follows. The task having the greatest processing time less than the "hiatus" is selected.

2. **COVERT**

   The flow chart for COVERT is Figure B.2. Entry is made only when there is at least one idle machine and at least one task in queue. The selected
GLOSSARY

LOPI: current lowest (most desirable) priority index
LOARPI: lowest index for arriving tasks
MAXPI: highest possible index
TWHEN: time at which slack is to be evaluated
T: current time
J: index at "standby job"
LONDX: index of current lowest priority index job
PT(I): processing time of task i
TEMP: temporary storage
U(I): urgent start date of task i
MAXT: maximum time
K: index of "alternate standby"
SLK(I): slack of task i
PI(I): priority index of task i
ROP(I): remaining operations for task i

\[
\begin{align*}
\text{ENTER} \\
\text{in hold-off?} \quad \text{Y} \quad \text{N} \\
\text{new arrival?} \quad \text{Y} \quad \text{N} \\
\text{EXIT}
\end{align*}
\]

\[
\begin{align*}
\text{LOPI} &= \text{MAXPI} \\
\text{LOARPI} &= \text{MAXPI} \\
\text{TWHEN} &= \text{T}
\end{align*}
\]

"OPERATION SELECTION"

\[
\begin{align*}
\text{J} &= \text{LONDX} \\
\text{TWHEN} &= \text{T} + \text{PT} (\text{J}) \\
\text{TEMP} &= \text{U} (\text{J}) \\
\text{U} (\text{J}) &= \text{MAXT}
\end{align*}
\]

"ALTERNATE OPERATION"

\[
\begin{align*}
\text{U} (\text{J}) &= \text{TEMP} \\
\text{K} &= \text{LONDX} \\
\text{PI} (\text{K}) &= \text{LOPI} \\
\text{TWHEN} &= \text{T} + \text{PT} (\text{K}) \\
\text{SLK} (\text{J}) &= \text{U} (\text{I}) - \text{TWHEN}
\end{align*}
\]

\[
\begin{align*}
\text{SLK} (\text{K}) &> 0 \\
\text{N} \\
\text{Y} \\
\text{PI} (\text{J}) &= \text{SLK} (\text{J}) \\
\text{PI} (\text{J}) &= \text{SLK} (\text{J}) \cdot \text{ROP} (\text{J}) \\
\text{PI} (\text{J}) &= \text{PI} (\text{K}) \\
\text{J} &= \text{K}
\end{align*}
\]

Figure B.1: Flow Chart for GERES.
GLOSSARY

FIN(I): finish time of task i
ARRT(I): arrival time of task i
H(K): "hiatus" of task i, time until arrival
SNKT: current largest feasible insert task processing time
SNK: insert indicator; 2 indicates an insertable task has been found
SNKDX: index of insertable task

Figure B.1; continued
Figure B.1; continued
task is indicated by index J. The "normal" start date shown in the flow chart has been adjusted for the particular "k" being used.

3. CTPLS (for single component orders, multiple channel machine groups)

This heuristic simply modifies COVERT, under certain circumstances, to be based on TAU, the time until the next machine comes free. A flow chart of the tests is shown in Figure B.3. The routine then proceeds to rank just as COVERT, except that TAU(I) is used instead of PT(I), except in breaking ties.

4. WORKS (for single component orders, single or multiple channel machine groups)

WORKS adds the heuristic for "hold-off" and "sneak-in" to that which has gone before. A threshold test is made on queue length to determine whether a hold off will be allowed. Ranking takes place pretty much as before except that time until arrival is added to processing time for arriving orders. If a critical, present task is selected, it is placed in its next assigned queue as well as on the current idle machine, so that it may be considered as a hold-off candidate at that queue.

If an arriving order is selected the hold-off analysis routine is entered. The tasks are considered in decreasing order of c/t and the time of start is determined first assuming that hold-off occurs. Sneak-ins are explicitly considered in priority order and the possibility of more than one sneak-in is considered. Then start times are computed assuming no hold-off (and no sneak-in). The difference in start times, and hence delay is costed out, and if it is cheaper to holdoff that option is taken. The flow chart is shown in Figure B.4.
GLOSSARY:

LOPI: current lowest (most desirable) priority index
MAXPI: largest possible index
MAXT: maximum time
LOPT: processing time for current selected task (to break ties)
U(I): urgent start date for task i
N(I): normal start date for task i
PT(I): processing time for task i
T: current time
CONS: a large constant
LONDX: index of current selected task

```
ENTER

LOPI = MAXPI
LOPT = MAXT

1

Search Q for task

J = LONDX

EXIT

no more in queue

U(I): T

N(I): T

Pi(I) = \[ \frac{U(I) - N(I)}{T - N(I)} \] * PT(I)

Pi(I): LOPI

Pt(I): LOPT

LONDX = I
LOPI = Pi(I)
LOPT = PT(I)
```

Figure B.2: Flow Chart for COVERT
GLOSSARY

G: machine group index
NM(G): number of machines in group G
NQ(G): number of tasks in queue, group G
LIM(G): Upper limit for using TAU modification
NIDLE(G): number of idle machine in G
FIN(G): time next machine becomes free in G
EPSIL: time until next machine becomes free in G

\[ \text{LOPI} = \text{MAXPI} \]
\[ \text{LOPT} :: \text{MAXT} \]

\[ \text{NM(G)} = 1 \]

\[ \text{NQ(G)} \leq \text{LIM(G)} \]

\[ \text{NIDLE(G)} = 1 \]

EPSIL = FIN(G) - T

Rank by COVERT using
TAU min (EPSIL, PT(I)) vice PT(I), except for breaking ties

Figure B.3: Flow Chart for CTPLS.
GLOSSARY

HOLIM(G): upper limit on \( NQ(G) \) for hold off to occur.

Note: rest of flow assumes "HO" is set; otherwise CTPLS is carried out, then transfer is made to connector 3.

![Flow Chart](image)

Figure B.4: Flow Chart for WORKS, single component orders, single or multiple channel machine groups.
GLOSSARY:
K: idle machine index
TWITH(I): time of performance assuming holdoff
TSNK: time when sneaked in task will start

Figure B.4, continued
"potential sneak-in"

\[
TWITH(I) = TSNK
\]

\[
TSNK = TSNK + PT(I)
\]

\[
HIATUS = HIATUS + PT(I)
\]

GLOSSARY

TCOST(I): time basis for cost rate applicable to difference between TWITH and TWO
TWO: start time assuming no hold off
C(I): cost rate for delay of task i
DELTAT: difference in delay with and without holdoff
TOTC: total difference in penalties.

Figure B.4; continued
Figure B.4; continued
5. WORKS (for multiple component orders, single or multiple channel machine groups)

An outline of the multiple component rule is provided by Figure B.5. The pattern is familiar except that the "c" is calculated for the latest branch on the order and the segment slack reflects the amount of delay that can be tolerated without increasing "c". Slack is used in the initial ranking only to settle ties.

After ranking, if the alternate machine option is being used, a search is made, task by task in priority order, for a machine to assign the task to, which assignment will not result in order delay. The latest finishing machine is selected for assignment. Any tasks for which alternate machines are found are excluded from consideration in evaluating the permutations. The alternate assignments are, however, considered in calculating the "TWHEN" for the tasks still under consideration.

The top-ranked four (or fewer) tasks are then simply laid out in all possible permutations. Each is costed by the logic shown, and the task which is first in the lowest cost permutation is selected. Of course, hold-off can be selected as well, if the conditions are met.
C(I): delay cost for latest branch on order

1

Test hold-off conditions

Search Q for task

no more in queue

alternate machine?

use alternate machine?

N

Y

calculate rank by min PT(I) + H(I)
C(I)
as before; break ties by minimum segment slack;
place tasks in array in priority sequence; max array sizes:
4 for single ch.
2 for 2 ch.
10 for 4 ch.

select task from top of array; if a machine finishes
before slack is exhausted, remove task from array
(update latest such machine.)
Continue for all tasks.

for each of the 24 permutations of the four top ranked
tasks, calculate the total delay cost.

Figure B.5: Outline of flow for WORKS; multiple component orders.
Item costing:

\[ SLKT(I) = T + SLK(I) + H(I) \]

\[ TCOST(I) = \frac{TWHEN(I) + SLKT(I)}{2} \]

\[ C(I) = \frac{TCOST(I) - N(I)}{U(I) - N(I)} \]

\[ \text{DELTAT} = TWHEN(I) - SLKT(I) \]

\[ \text{SUMC} = \text{SUMC} + \text{DELTAT} \times C(I) \]

\[ \text{GET NEXT TASK} \]
APPENDIX C

Logic for Updating Segment Slack

1. Upon arrival at queue (Figure C.1)

The principle of the slack accounting scheme employed here is that each time a task is subjected to a known delay, its segment slack is updated. If its current slack is greater than the delay, only slack is reduced. If delay is greater than slack, then a new critical path has been created, and all slacks are updated.

On arrival in queue, the known delay, if any, is determined. This may be the time until a machine next becomes free, or zero, if there is an idle machine. Distinction is made in the updating process depending on whether the order is already tardy, late, or early, and changes of status are recognized.

Two recompute subroutines, REDOCP and REDOCR, are employed when, either because of rounding error in the schedule dates, or status change, the entire order must be reevaluated. REDOCR updates orders which are already tardy (viz. Figure C.4). REDOCP contains the logic for orders which are not yet tardy, (viz. Figure C.3).

2. After a selection (Figure C.2)

The other updating time comes when a selection of one task has been made and the remaining tasks are then subjected to delay until the next machine in the group becomes free. The logic is generally the same.
GLOSSARY:

TWHEN: effective update time
LASTT(G): time rest of Q updated to.
EWT(i): expected remaining wait time for task i
C(O): estimated prob. of order tardiness.

Figure C.1: Flow Chart for Critical Path Update upon Queue Entry
Figure C.2: Flow Chart for Critical Path Update Upon Incurrence of Delay
Figure C.3: Flow Chart for REDOCP (for orders not yet tardy)
UNEXT: urgent start
date of next operation
FIN: completion time
do current operation

For each active
segment

LAT(I) = TWEN-U(I)
LAT(I) = UNEXT(I)-FIN(I)

LAT(I): MXLAT

MXLAT = LAT(I)
generate next
segment

SLK(I): MXLAT-LAT(I)
FOR ALL I

EXIT

Figure C.4: Flow Chart for REDOCR (for orders which are tardy)
APPENDIX D

Summary of Experimental Results

1. Table D.1: Experimental Results for Single Component Orders, Single Channel Machine Groups

2. Table D.2: Experimental Results for Single Component Orders, Multiple Channel Machine Groups and the "General Shop"

3. Table D.3: Experimental Results for Multiple Component Orders
## TABLE D.1
Experimental Results for Single Component Orders, Single Channel Shop (n = 3072)

<table>
<thead>
<tr>
<th>Rule</th>
<th>$U = .70$</th>
<th>$U = .80$</th>
<th>$U = .90$</th>
<th>$U = .80$</th>
<th>$U = .80$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal due dates</td>
<td>Normal due dates</td>
<td>Normal due dates</td>
<td>Loose due dates</td>
<td>Tight due dates</td>
</tr>
<tr>
<td>FCFSV</td>
<td>T 4.3 6.8</td>
<td>T 8.6 14.4</td>
<td>T 57.9 32.5</td>
<td>T 13.5 13.9</td>
<td>T 60.1 18.3</td>
</tr>
<tr>
<td></td>
<td>W 6.8 14.4</td>
<td>W 8.6 32.5</td>
<td>W 42.5 13.9</td>
<td>W 6.6 13.9</td>
<td>W 18.3 15.1</td>
</tr>
<tr>
<td>EARSID</td>
<td>T 6.6 6.6</td>
<td>T 9.8 14.2</td>
<td>T 38.6 32.7</td>
<td>T 4.5 3.1</td>
<td>T 13.8</td>
</tr>
<tr>
<td></td>
<td>W 6.6 14.2</td>
<td>W 9.8 32.7</td>
<td>W 30.3 13.8</td>
<td>W 3.1 13.8</td>
<td>W 14.8</td>
</tr>
<tr>
<td>SLROP</td>
<td>T 8.3 6.8</td>
<td>T 4.0 13.9</td>
<td>T 16.8 31.3</td>
<td>T 1.3 0.8</td>
<td>T 13.7</td>
</tr>
<tr>
<td></td>
<td>W 6.8 13.9</td>
<td>W 4.0 31.3</td>
<td>W 15.3 13.7</td>
<td>W 0.8 13.7</td>
<td>W 14.8</td>
</tr>
<tr>
<td>GERES</td>
<td>T 6.9 6.8</td>
<td>T 9.9 14.6</td>
<td>T 15.2 31.3</td>
<td>T 1.0 0.8</td>
<td>T 13.5</td>
</tr>
<tr>
<td></td>
<td>W 6.8 14.6</td>
<td>W 9.9 31.3</td>
<td>W 32.5 13.5</td>
<td>W 0.8 13.5</td>
<td>W 14.8</td>
</tr>
<tr>
<td>SHTOP</td>
<td>T 1.9 4.0</td>
<td>T 4.0 7.0</td>
<td>T 28.3 13.6</td>
<td>T 5.8 6.8</td>
<td>T 17.0</td>
</tr>
<tr>
<td></td>
<td>W 4.0 7.0</td>
<td>W 4.0 6.8</td>
<td>W 33.7 13.6</td>
<td>W 6.8 17.0</td>
<td>W 9.3</td>
</tr>
<tr>
<td>TRSIO</td>
<td>T 2.9 4.5</td>
<td>T 2.0 8.0</td>
<td>T 4.9 15.8</td>
<td>T 1.6 7.2</td>
<td>T 20.1</td>
</tr>
<tr>
<td></td>
<td>W 4.5 8.0</td>
<td>W 2.0 15.8</td>
<td>W 3.5 15.8</td>
<td>W 7.2 20.1</td>
<td>W 11.2</td>
</tr>
<tr>
<td>COVERT (K=0.5)</td>
<td>T 6.4 2.8</td>
<td>T 2.8 8.8</td>
<td>T 3.3 2.9</td>
<td>T 18.9</td>
<td>T ---</td>
</tr>
<tr>
<td></td>
<td>W 4.8 8.8</td>
<td>W 2.8 2.9</td>
<td>W 8.8 18.9</td>
<td>W ---</td>
<td>W ---</td>
</tr>
<tr>
<td></td>
<td>(K=1.0)</td>
<td>4.2 1.3</td>
<td>5.1</td>
<td>2.5 1.8</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T 2.0 10.3</td>
<td>W 4.7 5.1</td>
<td>T 10.3</td>
<td>T 21.3</td>
</tr>
<tr>
<td></td>
<td>(K=1.5)</td>
<td>3.8 1.6</td>
<td>5.0</td>
<td>2.8 1.9</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W 5.0 10.2</td>
<td>W 2.8 7.1</td>
<td>W 22.6</td>
<td>W 6.5 7.1</td>
</tr>
<tr>
<td></td>
<td>(K=2.0)</td>
<td>5.3 2.0</td>
<td>5.0</td>
<td>3.4 2.3</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T 5.0 10.0</td>
<td>W 3.4 2.3</td>
<td>T ---</td>
<td>T ---</td>
</tr>
<tr>
<td>WORKS (K=0.5)</td>
<td>T ---</td>
<td>T ---</td>
<td>T 1.3 1.2</td>
<td>T 8.9 1.4</td>
<td>T 17.5</td>
</tr>
<tr>
<td></td>
<td>(K=1.0)</td>
<td>4.0 1.5</td>
<td>5.0</td>
<td>2.0 1.5</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>(K=1.5)</td>
<td>4.0 1.5</td>
<td>5.0</td>
<td>2.0 1.5</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>(K=2.0)</td>
<td>4.5 1.9</td>
<td>5.0</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>
TABLE D.2
Experimental Results for Single Component Orders, Multiple Channel and General Shops

<table>
<thead>
<tr>
<th>Rule</th>
<th>2 machines/group, $U = .83$</th>
<th>4 machines/group, $U = .85$</th>
<th>General shop, $U = .84$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n = 2048$</td>
<td>$\bar{x}$</td>
<td>$\bar{w}$</td>
</tr>
<tr>
<td></td>
<td>$\bar{x}$</td>
<td>$\bar{w}$</td>
<td>$\bar{x}$</td>
</tr>
<tr>
<td>FCFSV</td>
<td>28.7</td>
<td>16.4</td>
<td>11.2</td>
</tr>
<tr>
<td>EARSD</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>SLROP</td>
<td>18.4</td>
<td>15.3</td>
<td>10.5</td>
</tr>
<tr>
<td>SHTOP</td>
<td>7.9</td>
<td>4.6</td>
<td>5.0</td>
</tr>
<tr>
<td>TRSIO</td>
<td>5.7</td>
<td>3.9</td>
<td>6.1</td>
</tr>
<tr>
<td>COVERT</td>
<td>3.1</td>
<td>3.8</td>
<td>7.3</td>
</tr>
<tr>
<td>CTPLS (N=2)</td>
<td>3.4</td>
<td>4.0</td>
<td>7.2</td>
</tr>
<tr>
<td>(N=4)</td>
<td>4.0</td>
<td>4.1</td>
<td>7.2</td>
</tr>
<tr>
<td>COTUTB (N=0)</td>
<td>3.3</td>
<td>3.9</td>
<td>7.3</td>
</tr>
<tr>
<td>(N=2)</td>
<td>3.2</td>
<td>3.3</td>
<td>7.1</td>
</tr>
<tr>
<td>(N=4)</td>
<td>4.4</td>
<td>3.5</td>
<td>7.2</td>
</tr>
<tr>
<td>WORKS (N=0)</td>
<td>2.9</td>
<td>3.0</td>
<td>7.2</td>
</tr>
<tr>
<td>(N=2)</td>
<td>3.1</td>
<td>3.3</td>
<td>7.3</td>
</tr>
<tr>
<td>(N=4)</td>
<td>3.7</td>
<td>4.2</td>
<td>7.4</td>
</tr>
<tr>
<td>COTADD</td>
<td>4.4</td>
<td>4.0</td>
<td>7.6</td>
</tr>
</tbody>
</table>
**TABLE D.3**

Experimental Results for Multiple Component Orders \((n = 2048)\)

<table>
<thead>
<tr>
<th>Rule</th>
<th>3 component orders</th>
<th>6 component orders</th>
<th>3 component orders</th>
<th>3 component orders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 machine/group, (U = .80)</td>
<td>1 machine/group, (U = .80)</td>
<td>2 machines/group, (U = .83)</td>
<td>2 machines/group, (U = .83)</td>
</tr>
<tr>
<td>(\overline{T})</td>
<td>(\overline{T})</td>
<td>(\overline{T})</td>
<td>(\overline{T})</td>
<td>(\overline{T})</td>
</tr>
<tr>
<td>FCFSV</td>
<td>40.5</td>
<td>15.4</td>
<td>12.7</td>
<td>28.5</td>
</tr>
<tr>
<td>EARSD</td>
<td>17.2</td>
<td>6.0</td>
<td>12.6</td>
<td>20.5</td>
</tr>
<tr>
<td>SLROP</td>
<td>5.9</td>
<td>3.2</td>
<td>12.5</td>
<td>24.0</td>
</tr>
<tr>
<td>SHTOP</td>
<td>16.2</td>
<td>2.9</td>
<td>5.9</td>
<td>29.4</td>
</tr>
<tr>
<td>TRSIO</td>
<td>6.3</td>
<td>2.2</td>
<td>6.4</td>
<td>30.4</td>
</tr>
<tr>
<td>COVERT</td>
<td>1.9</td>
<td>0.7</td>
<td>9.1</td>
<td>23.8</td>
</tr>
<tr>
<td>CTPLS</td>
<td>2.3</td>
<td>0.8</td>
<td>9.3</td>
<td>21.2</td>
</tr>
<tr>
<td>WORKS</td>
<td>1.4</td>
<td>0.4</td>
<td>9.3</td>
<td>21.4</td>
</tr>
<tr>
<td>alt. mach.</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>alt. mach., hold-offs</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY

Books


Articles, Reports, and Theses


Holt, Charles C. "Priority Rules for Minimizing the Cost of Queues in Machine Scheduling," Chapter 6 of Industrial Scheduling, 83-95.


Jackson, James R. "Queues with Dynamic Priority Disciplines," Chapter 19 of Industrial Scheduling, 308-326.


Johnson, Selmer. "Discussion" (of Mitten, see below), Management Science, V (April, 1958), 299-303.

Johnson, Selmer. "Optimal Two- and Three-Stage Production Schedules with Set-up Times Included." Chapter 2 of Industrial Scheduling, 13-20.


Biographical Note on the Author

Donald C. Carroll was born on November 5, 1930 in Durham, North Carolina. He is the son of Dudley D. and Eleanore E. Carroll of Chapel Hill, North Carolina. After attending public schools in Chapel Hill, he finished his college preparation at Westtown School, Westtown, Pa.

After serving for one year in the Marine Corps, he enrolled at the University of North Carolina, receiving a B.S. in mathematics in January 1954. He was elected to membership in Phi Beta Kappa, Pi Mu Epsilon, Delta Phi Alpha, and Phi Eta Sigma honorary fraternities and was a member of Alpha Tau Omega social fraternity.

Upon graduation, he served for two and one half years on active duty as a Marine officer, with foreign service in Korea and Japan. He resigned his commission as a captain in 1959.

He entered the School of Industrial Management at M.I.T. in 1956, receiving an S.M. in September 1958. His master's thesis, entitled "A Random-Demand Joint-Item Warehouse Inventory Control System" was published by the Operations Research Center. He was elected to membership in Tau Beta Pi in 1958, and served as secretary-treasurer of the Graduate Student Council that year.

This was followed by two years of industrial employment, the first in the Electronic Planning Department of Pittsburgh Plate Glass Company, in Pittsburgh, designing computer-based production and inventory control systems, the second at the Manufacturing Control Department of Westinghouse Electric Corporation, also in Pittsburgh. In the latter position, he acted as an internal consultant on the applications of computers in manufacturing control. His interest in job shop sequencing problems stems from this experience.
He returned to M.I.T. in 1960 as a Sloan Teaching Intern and was appointed assistant professor in 1961, his areas of teaching concentration being in production management, management information systems, and statistics. His promotion to associate professor becomes effective July 1 of this year.

He holds membership in the Institute of Management Sciences, the Association for Computing Machinery, and the American Institute of Industrial Engineers.

His wife is the former Lucy Anne James of Newton, Connecticut; they have two children, Curtis James and Leah Anne and live in Lexington, Massachusetts.