DYNAMIC ROUTING IN AN
UNRELIABLE MANUFACTURING NETWORK
WITH LIMITED STORAGE

by

Ellen Louise Hahne

This report is based on the unaltered thesis of Ellen Louise Hahne, submitted in partial fulfillment of the requirements for the degree of Master of Science at the Massachusetts Institute of Technology in February 1981. The research was carried out at the Laboratory for Information and Decision Systems with partial support extended by the National Science Foundation under grant DAR78-17826.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author, and do not necessarily reflect the views of the National Science Foundation.

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
The author hereby grants to:

Bell Telephone Laboratories, Incorporated,
the Massachusetts Institute of Technology,
the Laboratory for Information and Decision Systems,
the National Science Foundation,
and the National Technical Information Service

permission to reproduce and to publicly distribute
copies of this thesis document in whole or in part.
DYNAMIC ROUTING IN AN
UNRELIABLE MANUFACTURING NETWORK
WITH LIMITED STORAGE

by

ELLEN LOUISE HAHNE

Submitted to the Department of Electrical Engineering
and Computer Science on January 13, 1981
in partial fulfillment of the requirements
for the Degree of Master of Science

ABSTRACT

Optimal dynamic routing strategies are computed for
a simple manufacturing network with unreliable machines
and finite storage buffers. A Markov process model of
the system incorporates random processing times and ran-
don failures and repairs. Routing policies which
maximize system throughput are computed using a vari-
tion of D. J. White's method of successive approxima-
tions. The optimal routing rule and several heuristics
are compared with regard to structure and performance
over wide ranges of the system parameters.

Thesis Supervisor:  Dr. Stanley B. Gershwin
Title:  Lecturer, Department of Electrical Engineering
        and Computer Science
        Principal Research Scientist, Laboratory for
        Information and Decision Systems

Thesis Supervisor:  Dr. Michael Athans
Title:  Professor of Systems Science and Engineering
        Director, Laboratory for Information and
        Decision Systems
ACKNOWLEDGMENTS

The author's educational program at M.I.T. has been sponsored by Bell Laboratories. The extraordinary generosity of this support is gratefully acknowledged.

The thesis research was conducted in the Laboratory for Information and Decision Systems at M.I.T., with partial support extended by the National Science Foundation under grant DAR76-17826.*

Acknowledgments are due to Prof. Michael Athans and Dr. Stanley Gershwin for suggesting this thesis topic, supervising the research, and reviewing the manuscript. Special thanks go to Dr. Gershwin for the very generous commitment of his time and for countless suggestions which have had a major impact on this work. Others whose input and assistance have been most helpful include Messrs. Mostara Ammar, Isidro Castineyra, Orlando Sotomayor-Diaz, and Joseph Kimemia, Miss Karen Randell, Drs. John Palmer and David Kettler, and Profs. Loren Platman, Dimitri Bertsekas, Alvin Drake, and Richard Larson.

The author would also like to thank her faculty counselor, Prof. Alvin Drake, and her supervisors at Bell Laboratories, Mr. Detlev Haenschke and Dr. David Kettler, for their constant personal and administrative support.

Credit for the drafting goes to Mr. Arthur Giordani and for the typing to Mrs. Helen Colodin. This document attests to their talented and dedicated efforts.

This research would have been economically infeasible if not for the extremely low evening rates charged by the Information Processing Center at M.I.T. This student-oriented pricing policy is greatly appreciated.

* Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.
To My Family
# TABLE OF CONTENTS

**ABSTRACT**

3

**ACKNOWLEDGMENTS**

4

**DEDICATION**

5

**TABLE OF CONTENTS**

6

1. **INTRODUCTION**

1.1 Problem Description and Motivation

10

1.2 Review of Related Research

14

1.3 Contribution of this Thesis

34

1.4 Organization of this Report

35

2. **MODEL FORMULATION AND FORMAL PROBLEM STATEMENT**

37

2.1 Physical Description of the System Model

38

2.1.1 Network Layout and Material Flow

38

2.1.2 Workpieces

39

2.1.3 Storage

40

2.1.4 Machines

42

2.1.5 System Parameters and System State

49

2.1.6 Control Objective and Constraints

50

2.2 Formulation as a Continuous-Time Markov Decision Process

52

2.2.1 State Space Description

52

2.2.2 State Transitions

56
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.3 Steady-State Probabilities</td>
<td>63</td>
</tr>
<tr>
<td>2.2.4 Rewards</td>
<td>64</td>
</tr>
<tr>
<td>2.2.5 Optimization Objective</td>
<td>65</td>
</tr>
<tr>
<td>2.3 An Equivalent Discrete-Time Model</td>
<td>66</td>
</tr>
<tr>
<td>2.4 Howard's Dynamic Programming Formulation</td>
<td>72</td>
</tr>
<tr>
<td>2.5 Summary</td>
<td>78</td>
</tr>
<tr>
<td>3. DESCRIPTION OF THE ALGORITHM</td>
<td>79</td>
</tr>
<tr>
<td>3.1 White's Method of Successive Approximations</td>
<td>79</td>
</tr>
<tr>
<td>3.2 Time Scale Selection</td>
<td>82</td>
</tr>
<tr>
<td>3.3 Acceleration Procedure</td>
<td>86</td>
</tr>
<tr>
<td>3.4 Successive Substitution Procedure</td>
<td>88</td>
</tr>
<tr>
<td>3.5 Algorithm Actually Implemented</td>
<td>89</td>
</tr>
<tr>
<td>3.6 Convergence Criterion</td>
<td>93</td>
</tr>
<tr>
<td>3.7 Computational Complexity</td>
<td>97</td>
</tr>
<tr>
<td>3.8 Summary</td>
<td>99</td>
</tr>
<tr>
<td>4. A HEURISTIC FEEDBACK STRATEGY</td>
<td>100</td>
</tr>
<tr>
<td>4.1 The Proposed Heuristic: Worktime Balance</td>
<td>100</td>
</tr>
<tr>
<td>4.2 Mathematical Statement of the Heuristic</td>
<td>105</td>
</tr>
<tr>
<td>4.2.1 Rule for Reliable Downstream Machines</td>
<td>105</td>
</tr>
<tr>
<td>4.2.2 Rule for Unreliable Downstream Machines</td>
<td>109</td>
</tr>
<tr>
<td>4.3 Graphical Description of the Heuristic</td>
<td>114</td>
</tr>
<tr>
<td>4.4 Summary</td>
<td>121</td>
</tr>
</tbody>
</table>
5. STRUCTURE OF THE OPTIMAL ROUTING POLICY

5.1 Properties of the Optimal Strategy for All Models

5.2 Simplest Model: All Machines Reliable
   5.2.1 General Shape of the Decision Boundary
   5.2.2 Intercept of the Decision Boundary
   5.2.3 Asymptotic Slope of the Decision Boundary

5.3 Intermediate Model: Unreliable Lead Machine and Reliable Downstream Machines

5.4 Complex Model: All Machines Unreliable

5.5 Summary

6. PERFORMANCE OF VARIOUS ROUTING STRATEGIES

6.1 Outline of this Study

6.2 Effect of Buffer Capacity

6.3 Effect of System Balance

6.4 Effect of Lead Machine Reliability

6.5 Summary

7. SUGGESTIONS FOR FUTURE RESEARCH

7.1 Remaining Work on this Problem

7.2 Variations and Extensions of this Problem

7.3 Stochastic Dynamic Scheduling Problems

7.4 Considerations of State Space Size

8. SUMMARY AND CONCLUSIONS
APPENDIX 1. STRUCTURE OF THE TRANSITION RATE MATRIX 196

APPENDIX 2. ANALYSIS OF PART COMPLETION TIME AT AN UNRELIABLE MACHINE 205

APPENDIX 3. THE SHORT-TERM ROUTING STRATEGY 217

APPENDIX 4. A ROUTING STRATEGY BASED ON STARVATION PROBABILITIES 227

REFERENCES 231
1. INTRODUCTION

1.1 Problem Description and Motivation

The routing of items through congested networks is a problem which arises in many contexts. Important examples include the routing of packets in data communication networks (Frank and Chou [1971], Foschini [1977], Foschini and Salz [1978]), the assignment of tasks in multiprocessor computer systems (Chow and Kohler [1976, 1977]), the control of vehicles in transportation networks with restricted-access highways (Shaw [1972, 1976], Schwartz and Tan [1977]), and the distribution of workpieces to machine tools in metalcutting shops (Solberg [1976]). Each example mentioned includes a routing controller which attempts to keep the overall system running smoothly. Due to random disturbances, such as variations in the time required to service items, the routing controller may not accurately know the current state of the system, nor can it predict with certainty the future of the system and the effect of routing decisions. Improved performance should therefore be possible if the controller can also observe the network and base the routing decisions on its current state. Such feedback control, called dynamic routing, allows items to be diverted from areas of extreme congestion and
eliminates unnecessary idleness of processors. This can reduce the time items spend in the system, increase system throughput, and improve the utilization of system resources.

Feedback is especially important if the system components are unreliable; if the controller knows when a component has failed, it can route items along other available paths. Dynamic routing is also particularly helpful if there is only limited queueing space for items which cannot be processed immediately. This is especially true if the storage space is distributed throughout the network. Such restrictions exacerbate the congestion problem and make efficient routing critical, since items may be refused admittance into the system when storage areas are full, and processors may be forced to stop when adjacent storages are full or empty.

This thesis is a study of dynamic routing strategies for the simplest system exhibiting the key features just mentioned: random failures and limited, distributed storage. The primary application we have in mind is the routing of workpieces in a fully automated manufacturing system, so the terminology and some of the assumptions of our model reflect this. Figure 1.1 shows the layout of the two-way branching network to be
Figure 1.1. Layout of the manufacturing network.
studied. The squares $M_0$, $M_1$ and $M_2$ represent unreliable workstations called machines. The circles $B_1$ and $B_2$ represent finite storage areas called buffers. Discrete workpieces, also called parts, move through the system as indicated by the arrows. Each part must first be processed by the lead machine $M_0$. The part may then be processed by either downstream machine $M_1$ or $M_2$. This routing decision is made after a part is processed at $M_0$ and is based on the state of the system at that time. (Such an arrangement of parallel, functionally equivalent downstream machines is appropriate if the individual downstream machines are very unreliable or if they are much slower than the lead machine.) Our measure of performance will be the system production rate or throughout, that is, the average number of parts produced per unit time. By modeling the system as a controlled Markov process, it is possible to numerically compute closed-loop routing rules which optimize system performance.

The goal of the research reported here is to examine the structure and performance of the optimal feedback routing strategy for this system. We investigate the effects of the machine speeds, the failure and repair rates, and the buffer capacities on the optimal
Several heuristic routing rules are studied as well and are compared with the optimal rule for two reasons. First, the suboptimal rules are based on intuition, so comparing the optimal solution against them aids in developing an understanding of the optimal functioning of the system. This in turn could lead to conjectures about the optimal functioning of similar systems, perhaps more complex ones. Second, such comparisons are also useful for determining whether easily computable feedback heuristics can be used instead of a truly optimal strategy with little sacrifice in performance, and when even open-loop control is satisfactory.

This thesis is also a test of the discrete state variable models and the dynamic programming procedures used to study the problem described above. We attempt to judge the applicability of such models and methods to other similar problems and the feasibility of extending the approach to larger and more complex problems.

1.2 Review of Related Research

The problem addressed in this thesis falls in the general category of optimal control of queues. Survey papers in this area include Bhat [1969], Crabill, Gross

More specifically, we are concerned with the assignment of randomly arriving customers to parallel servers, each with its own queue. To see this, consider the output of the lead machine \( M_0 \) in figure 1.1 as a stream of arrivals to the downstream portion of the network. If \( M_0 \) is reliable and requires an exponentially distributed amount of time to complete one part, then this arrival process is Poisson.

A great deal of work has been done on problems of this type. Most researchers, however, deal with randomized assignment strategies which do not consider the state of the system, i.e., open-loop strategies. Examples include Çınlar [1967] and Lemoine [1975]. Jackson [1957, 1963] pioneered the study of queueing networks with generalized topologies. Such "Jackson network" models always assume open-loop, randomized assignment of customers to queues. Lemoine [1977] and Disney [1975] survey the research in this area.

Of the many papers analyzing systems with open-loop assignment strategies, we discuss here only those involving optimal routing to parallel queues. Buzen and
Chen have devoted their efforts to such problems (Buzen [1971 a, 1971 b, 1973], Chen [1973 a,b,c], Buzen and Chen [1974], Buzen and Shum [1974]). Chen concentrates on an open queueing network, that is, one with an infinite source of customers and infinite buffers. His objective is to minimize the time an average customer spends in the system. Chen develops algorithms to find the optimal randomized split of a Poisson input stream among the various servers, nearly achieving closed-form solutions. Servers are, of course, heterogeneous, with exponentially distributed service times in Chen [1973 a, b] and general service time distributions in Chen [1973 c] and Buzen and Chen [1974]. More complex objective functions are also considered in Chen [1973 b].

Buzen uses Chen's method to determine the optimal routing split in an open network for a very specialized server model in Buzen [1973] and Buzen and Chen [1974]. Most of Buzen's work, however, deals with a closed queueing network, that is, one with a finite number of circulating customers and buffers each large enough to hold all customers (Buzen [1971 a,b], Buzen and Shum [1974]). His model includes a central server (corresponding to the lead machine $M_0$ in our model) and
several parallel servers (corresponding to our downstream machines $M_1$ and $M_2$). Each server has its own input buffer. After visiting the central server, each customer is randomly routed to one of the parallel servers. The customer then loops back to the central server, thus circling through the network forever. Buzen's objective is usually to determine the randomized routing split to the parallel servers which maximizes system throughput, though the minimization of customer delay is also treated in Buzen and Shum [1974]. The closed network models also allow heterogeneous servers, but the processing times at any one server must be exponentially distributed. Buzen develops efficient algorithms for closed queueing networks but no closed-form solutions. The optimal open-loop randomized strategies for open networks and closed networks are compared in Buzen and Shum [1974].

Both Chen [1973 a] (studying the open network) and Buzen [1971 a] (studying the closed network) are surprised to find that optimal, randomized, open-loop routing to exponential servers does not equalize the utilizations of the servers or equalize the average queue lengths. Rather, the faster servers should be better utilized and should have longer average queue
lengths than the slower servers. This result is consistent with our findings in this thesis. As reported in chapter 5, optimal closed-loop routing to reliable, exponential servers tends to dynamically balance the queue lengths in a ratio which is between 1:1 and the ratio of the server rates. (This is a very rough description; see chapter 5 for more details.)

Other authors have also studied optimal open-loop routing. Foschini [1977] treats the special case of Chen's open network near saturation, that is, as the arrival rate of customers approaches the maximum processing rate of the system. Foschini determines the randomized routing split of a Poisson arrival stream to exponential servers which minimizes average customer delay.

Ushakov and Chernyshev [1976] generalize Chen's open network model by including a large number of customer classes. Service time distributions are general and may depend on customer class. The routing flexibility of the system is somewhat restricted, however: each customer class is (statically) assigned to a single server. Ushakov and Chernyshev propose a branch-and-bound algorithm to distribute the various arrival streams to servers in a way which minimizes the average
customer delay.

Rolfe [1968] studies a network of parallel, heterogeneous, multi-server facilities, each with its own customer stream. A customer who arrives to find all servers busy at his own facility may join the queue at that facility or leave the system or select another facility with idle servers. The system earns rewards for serving customers and incurs penalties for keeping customers waiting. Rolfe formulates the closed-loop routing optimization problem, which he cannot solve, so he concentrates on open-loop randomized versions of the problem. One version permits no feedback whatsoever, while the other model allows a customer to check the status of his own facility before routing elsewhere. Rolfe addresses many special cases, varying the number of servers, the service time distributions, and the maximum allowable queue lengths. He attempts to determine optimal routing rules for these cases, with varying degrees of success.

Ephremides, Varaiya and Walrand [1980] study open-loop routing to parallel, single-server queues. Service time distributions are identical and exponential, and the objective is to minimize the time to service a certain number of customers. Ephremides et al. show that a
round robin (i.e., alternating) policy is optimal provided that the initial queue lengths are known to be equal. They stress that the average customer delay in a system operated with the round robin rule is better than for a randomized split. Their study is exceptional because they do not assume Poisson arrivals. In fact, they show that any knowledge of the arrival process is irrelevant to the routing decision.

Another class of papers concerns state-dependent assignment of customers to servers. Hall and Disney [1971] set up a very general model of such a system with multiple types of customers arriving according to a semi-Markov process, with heterogeneous, exponential servers, and with finite buffers. Most authors, however, restrict their attention to a single customer class, Poisson arrivals, and exponential servers. Feedback routing rules are usually selected a priori and the system is analyzed under those rules; typically, no optimization is attempted. Schwartz [1974] studies "lane selection" models with several customer classes, where an arriving customer joins the queue of the server dedicated to that customer's class, except when the server for another class is idle. Liu and Kettler [1976] study a similar model with multi-server facili-
ties and a limit on the number of "foreign" customers each facility may serve at one time. These models of Schwartz and Liu and Kettler are very similar to that of Rolle [1968] discussed earlier.

Chen [1975] takes a novel approach to the inherent difficulty of closed-loop routing problems. First he models a multiprogrammed computer system by a central-server, closed queueing network with state-dependent routing probabilities. Then, since he cannot analyze such a model, he devises an algorithm which fine tunes the routing probabilities for an open-loop strategy until the model under open-loop control approximates the closed-loop version.

Many authors have analyzed systems with parallel, identical, exponential servers, each with its own buffer, where customers of a single class arrive in a Poisson stream and join the shortest queue (Haight [1958], Kingman [1961], Flatto and McKean [1977], Winston [1975, 1977b], Chow and Kohler [1976, 1977], Foschini [1977], Foschini and Salz [1978], and Ephremides, Varaiya and Walrand [1980]). Many of these researchers stress the substantial reduction in customer delay which is possible through the use of dynamic assignment strategies based on feedback of the queue lengths. Haight
[1958] and Foschini [1977] also treat the case of heterogeneous exponential servers. In Haight's model, customers join the shortest queue regardless of the server, while in Foschini's model, the customers are smarter: they scale the queue lengths by the respective service rates before making their decision.

Haight [1958] also examines this system when customers "jockey" (i.e., switch) from the longer queue to the shorter one. Other authors who analyze jockeying situations include Koenigsberg [1966], Disney and Mitchell [1970], and Chow and Kohler [1977]. Haight, Koenigsberg and Disney and Mitchell model heterogeneous servers; Chow and Kohler have identical processors.

While Chow and Kohler [1976, 1977] do compare the system throughput and the average delay per customer under several different operating strategies, only two of the studies mentioned so far actually optimize a state-dependent assignment strategy for a parallel-queue system. Under simple and similar cost criteria, both Winston [1975, 1977 b] and Ephremides et al. [1980] prove that the optimal operating policy for a system with homogeneous customers and parallel, identical, exponential servers is to assign arriving customers to the shortest queue. Ephremides et al. do not require
the assumption of Poisson arrivals; they show that any knowledge or the arrival process is irrelevant to the routing decision. Winston does assume Poisson arrivals, but manages to prove optimality in a very strong sense (that of stochastic order).

Two other studies of optimal feedback routing which are very similar to this thesis are those of Ireland and Thomas [1972] and Tsitsiklis [1980]. Ireland and Thomas consider exponential, heterogeneous servers, each with an infinite, first-in-first-out buffer. Multiple classes of customers arrive according to independent Poisson processes and are assigned to queues on the basis of the current queue lengths. The mean service time varies from server to server, but does not depend on the customer class. The system incurs a waiting cost per customer per unit time which varies with the customer class. Ireland and Thomas perform a finite-stage optimization; their algorithm computes routing policies which minimize the expected cost of serving the next K customers, where K is fixed. Their work is more general than ours in that they consider the interactions of competing customer classes. It is less general in that the problems of failures and limited queueing capacity are not addressed. It differs from our study in the choice
of the planning horizon: we optimize over the infinite horizon. The computed example in their paper shows an optimal routing policy for the preferred customer class which closely resembles the optimal strategies found in our research.

Tsitsiklis [1980] is currently studying optimal feedback routing for a model which is identical to ours except that continuous material, rather than discrete parts, flows through the system. A routing controller divides the flow between the two downstream branches of the network based on the current system state. Random failures and repairs and finite buffers are featured in this model. Service occurs at a deterministic rate when a machine is functioning. Tsitsiklis' study is of special interest, because it separates the influence of random failures and repairs on the optimal dynamic routing strategy from the effect of random processing times on that strategy. Such a distinction is difficult to make using the model of this thesis.

Several authors have also considered the optimal assignment of heterogeneous customers to heterogeneous servers where individual queues are not present. Either no queueing space is available or a single queue is shared by all servers. Routing decisions are required
when a customer arrives to find two or more servers idle. Such models are studied by Albright [1974, 1977], Winston [1975, 1977 a], Johansen [1975], and Seth [1977]. Winston and Johansen also permit reallocation of all customers in service whenever a customer leaves the system or a new customer arrives.

Finally, we mention an unusual paper by Noetzel [1977] which treats the complementary problem -- the optimal dynamic distribution of a large number of servers among a small number of customer queues. For a closed queueing network model with homogeneous customers and identical servers, Noetzel finds that system throughput is maximized by distributing the service capacity in proportion to the queue lengths. This result is most interesting, since it is analogous to a heuristic strategy for assigning customers to servers which is presented in chapter 4 of this thesis.

This completes our survey of the literature on the assignment of customers to parallel queues. Those authors whose work most closely resembles ours are Chen [1973 a], Buzen [1971 a], Ireland and Thomas [1972], and Tsitsiklis [1980]. All of these researchers find optimal routing strategies. Only Ireland and Thomas and Tsitsiklis study dynamic strategies. Only Buzen and
Tsitsiklis consider blocking due to limited storage. Only Tsitsiklis models random failures and repairs.

We will now discuss another queueing problem which is related to this thesis: the scheduling of customers or various classes for service at a single processor. First let us demonstrate the connection with our model of figure 1.1. We stated in section 1.1 that routing decisions are made after a part is processed by the lead machine $M_0$. For the moment, let us assume instead that the routing decision is made before processing begins. Such an assumption is appropriate if $M_0$ treats parts differently depending on their downstream destination. This altered model is also suitable for a system producing two types of parts, where one type must be routed to $M_1$, the other type to $M_2$, and where $M_0$ must select which type or part to process next.

It is possible to convert this altered version of our routing problem to an equivalent dual problem of customer selection by a single server, using the part/hole transformation of Ammar [1980] and Ammar and Gershwin [1980]. As parts move from left to right in figure 1.1, consider the complementary motion of empty spaces (holes) from right to left. It is important to realize that this concept of holes only makes sense for
systems with finite buffers. When \( M_0 \) chooses a downstream route for a part and deposits it in the appropriate buffer, this is equivalent to \( M_0 \) selecting a hole from one buffer or the other and removing it. Looking at the other end of the network, we see that if machines \( M_1 \) and \( M_2 \) are reliable and have exponentially distributed processing times, then the arrival of holes to the buffers through \( M_1 \) and \( M_2 \) (corresponding to the departure of parts from the buffers through \( M_1 \) and \( M_2 \)) is Poisson. Whenever a buffer is full of holes (i.e., empty of parts) this process is temporarily halted. Focusing on the holes as the entities of interest, then, we have the equivalent problem diagrammed in figure 1.2. \( M_0 \) must select which of two types of customers to serve next, in order to maximize system throughput. (The equivalence of these two problems has not been proven, but it is a very strong conjecture. See Ammar [1980] and Ammar and Gershwin [1980] for equivalence proofs for queueing models similar to this one.)

Much research has been done on this dual problem of customer selection by servers. Most of the work deals with priority queueing models. In such systems, the customer classes are ranked, and a customer of low priority is not selected when customers of higher priori-
Figure 1.2. Equivalent customer selection problem.
ity are waiting. Customer arrivals are almost always Poisson, with different rates for different priority classes. In most models, each customer class has its own queue. Service time distributions are usually general and can vary with customer class. Both finite and infinite buffer models have been studied. Countless variations of priority schemes are possible, many of which are discussed in detail in surveys by Jaiswal [1968] and Conway, Maxwell and Miller [1967]. The customer selection rules in priority queueing studies are usually fixed, though some models have a few variable parameters which permit a minor degree of optimization. A review of the literature on optimization of priority queues can be found in Crabill, Gross and Magazine [1973, 1977].

The scheduling decisions in priority queueing models are generally based on feedback of very little state information. Typically, decisions depend on the presence of any number of customers of the various priority classes and possibly on the priority class of the customer currently in service. The most dynamic models do permit decisions based on the length of time customers have been waiting (Jaiswal [1968, pp. 196-204], Jackson [1960, 1961, 1962], Kleinrock [1964].
Kleinrock and Finkelstein [1967]), but decisions based on queue lengths are rarely allowed. Two exceptions in which queue lengths do play a part in the scheduling of customers are Balachandran [1971] and Bell [1973]. Neither of their models, however, corresponds exactly to the dual of our routing problem.

The question of direct interest to us here — the optimal, dynamic selection of customers from parallel queues based on feedback of the queue lengths — has been addressed in the literature. Studies of this problem have been performed by Little [1961 b], Kakalik and Little [1971], Klimov [1974], Harrison [1975], Tcha and Pliska [1977], Robinson [1978] and Reed [1975]. (Related works include Schrage [1968] and Cox and Smith [1951]. Torbett and Harrison [1974] describe an application of Harrison's results.) In these studies, each customer class has its own infinite capacity buffer. Arrival rates, service rates, and system costs may depend on the customer class. For various system models and cost structures, these authors prove that a simple priority queueing discipline is optimal among all customer selection rules which are based on feedback of the queue lengths. That is, the only information used in making the optimal selection is the presence or absence
of customers in each queue. The length of a queue (unless it is zero) has no effect on the decision.

This striking result points out the critical nature of our assumption of finite buffers. For if this result were true for finite buffer systems (which it is not), then the optimal strategy for our original problem (figure 1.1) would be to route the output of the lead machine to only one of the buffers, even when the other buffer is empty. Only when the preferred buffer is full would a part be routed to the low-priority buffer. Such a strategy is obviously poor. In fact, the optimal dynamic routing rule does not resemble such a priority strategy even for very large buffer capacities. (See chapter 5 of this thesis.)

There are several customer selection studies which do not fit into the mold of priority queues. Bartholdi and Rosenthal [1979] find schedules to minimize customer delay for a completely deterministic model. Segal [1979] examines a dynamic customer selection rule which attempts to maintain the queue levels in a fixed ratio. The decisions are randomized, but the routing probabilities are functions of the ratio of the queue lengths at any time. Segal's concern is with customer delay rather than system throughput, so the problems of finite
buffers are not explicitly addressed. No optimization is done in his study.

Santana and Platzman [1979] are among the very few researchers in the area of customer selection who model finite buffers and deal directly with the problem of lost customers due to lack of queueing capacity. For a rather general (but completely reliable) system model, they devise an efficient algorithm to compute closed-loop customer selection rules which are optimal over a restricted class of such rules called "regular policies"; this class is much broader than the set of priority queueing rules. The dissertation of Santana [1980] may include some numerical studies of the customer selection problem using this algorithm, but it has not yet been published. It would be very interesting to compare such results with the findings of this thesis.

It seems that little research has been done on the optimal selection of customers from queues which sheds light on our dual problem of optimally assigning customers to queues. The principal difficulty is that most research in customer selection is concerned with customer delay rather than system throughput, and thus it is not unreasonable to use models with unlimited storage. However, the assumption of infinite buffers destroys the
equivalence of the two problems, as explained earlier. We conclude that the work of Santana and Platzman [1979] is the only customer selection research which bears directly on the routing problem of this thesis.

As we have seen, the problems of finite storage and processor failures have received little attention in the literature on routing and customer selection in queueing networks. These issues are central, however, in the literature on transfer lines (i.e., production systems which are serial networks of processors and buffers). In fact, the system model used in this thesis is a direct extension of a model studied by Gershwin and Berman [1978, 1979] for an unreliable, two-machine transfer line with a single finite buffer and random processing times. Berman [1979] extends this model to include a broader class of service time distributions, and the three-machine, two-buffer version of the model is currently being analyzed by Wiley [1981]. A very similar transfer line model is treated by Buzacott [1972]. Surveys of transfer line models with unreliable machines and finite buffers can be found in Gershwin and Berman [1979] and Buzacott and Hanifin [1978].
1.3 Contribution of this Thesis

This thesis is novel in that it deals with optimal dynamic strategies in systems with finite buffers and unreliable processors. This study is not simply a description or an analysis of a system operated under a heuristic strategy, as most of the routing research to date has been. The optimization is not over a limited class of decision rules, such as open-loop strategies or priority queueing rules or the regular policies of Santana and Platzman [1979], but over all non-randomized feedback laws. Moreover, unlike the optimal routing rule for the perfectly symmetrical system of Winston [1975, 1977 b] and Ephremides et al. [1980], the answer to the problem addressed here is far from obvious.

Since this research concerns system throughput rather than customer delay, the problem of losses due to finite buffers is directly addressed. Finally, the author is aware of only one other study (Tsitsiklis [1980]) of routing or customer selection which deals explicitly with unreliable arrival or service processes, though a few papers do assume sufficiently general service times and/or customer interarrival times that random failures could be included. While the effect of storage limitations and processor failures on throughput
has received much attention in the study of transfer lines, this thesis, like that of Tsitsiklis, goes a step further: it explores the potential of dynamic routing in parallel networks to reduce these ill effects.

The price paid for all the interesting features of this thesis is that an analytical treatment appears to be impossible. As is the case for other models of similar complexity (e.g., Ireland and Thomas [1972], Tsitsiklis [1980], and Santana and Platzman [1979]), numerical solutions must suffice for our problem. Unfortunately, this is a very high price to pay, for the computational burden is great.

1.4 Organization of this Report

The manufacturing system described in section 1.1 is modeled as a Markov decision process in chapter 2, and the dynamic programming formulation of the problem is given. Chapter 3 describes a version of D. J. White's successive approximations algorithm which is used to compute optimal routing rules. The principal heuristic strategy, called the worktime balance rule, is developed in chapter 4. The structure of the optimal feedback control is examined and compared with various heuristics in chapter 5, and performance comparisons of
these strategies appear in chapter 6. Chapter 7 outlines future research areas, and chapter 8 summarizes and concludes the thesis.
2. MODEL FORMULATION AND FORMAL PROBLEM STATEMENT

A detailed model of the manufacturing network is presented in this chapter, and the problem of determining a dynamic routing strategy which maximizes the throughput of the system is formally stated. Section 2.1 describes the modeling assumptions and the control objective in physical terms. A continuous-time Markov decision model of the manufacturing system is then constructed in section 2.2, and the optimization is expressed in terms of this model. Since existing algorithms for solving Markov decision problems are designed for discrete-time systems, the model of section 2.2 is converted to an equivalent discrete-time version in section 2.3. Finally, the optimization is reformulated as a dynamic programming problem in section 2.4.

This chapter borrows heavily from Gershwin and Berman [1978, 1979] for their Markov models of unreliable manufacturing systems and from Howard [1960] for his decision process notation, the discrete-time system equivalence, and the dynamic programming formulation.
2.1 Physical Description of the System Model

In this section we present the physical assumptions of the manufacturing system model used in this thesis. The model is a direct extension of the two-machine, unreliable transfer line model of Gershwin and Berman [1978, 1979] to a branching, three-machine network with routing flexibility. Most of the modeling assumptions and concepts, terminology and notation of this section are from Gershwin and Berman.

2.1.1 Network Layout and Material Flow

The manufacturing network to be studied is modeled as a network of three machines $M_0$, $M_1$ and $M_2$ and two storage buffers $B_1$ and $B_2$. Figure 1.1 shows the layout of this network. Discrete workpieces (also called parts) move through the system as indicated by the arrows in the figure. Each part is first processed at the lead machine $M_0$, then an immediate decision is made to route the part to one of the downstream machines $M_1$ or $M_2$. Parts which are assigned to $M_1$ wait in buffer $B_1$ until $M_1$ is available, and parts assigned to $M_2$ wait in buffer $B_2$. Once a part is placed in a buffer, the routing decision may not be changed. Finished parts leave the system after being processed by $M_1$ or $M_2$. 
2.1.2 Workpieces

All workpieces require two operations. The first must be performed by the lead machine $M_0$, and the second can be performed by either downstream machine, $M_1$ or $M_2$. The treatment a part receives at $M_0$ does not depend on its downstream destination, so the routing decision is not made until processing at $M_0$ is completed.

It is assumed that workpieces are not identical and that processing times vary considerably from part to part. Such variations could be due to differences in workpiece size, material composition, or complexity of the required operations. We assume, however, that exact processing times for each workpiece are not known in advance, or that it is too inconvenient to store and use this information for making routing decisions. The routing controller knows only the statistical distribution of processing times at each machine. Thus the time required to perform an operation is modeled as a random variable whose distribution depends only on the machine performing the operation. (Further details are given in section 2.1.4.) Processing times for all operations are assumed to be independent. In particular, the processing time of a part at a downstream machine is independent of its processing time at the lead machine. Also,
the order in which workpieces enter the system and the order in which they are selected from buffers has nothing to do with their processing times.

Though machines may fail, workpieces are never damaged or rejected in this model. All completed parts are considered equally valuable, regardless of any machine failures experienced, regardless of the processing time or time spent in storage, and regardless of the route taken through the network.

2.1.3 Storage

The two internal storage areas $B_1$ and $B_2$ have limited capacities. The number of workpieces in $B_1$, including any part being processed in machine $M_1$, may not exceed $N_1$. Similarly, the number of parts in $B_2$, plus a possible part in $M_2$, may not exceed $N_2$. We call $N_1$ and $N_2$ the buffer capacities.

When a buffer is empty but the corresponding downstream machine holds a part, the machine will process that part. When the buffer and downstream machine are both empty, then the machine stops; it is said to be starved. When one buffer is full, the lead machine $M_0$ continues to work, placing completed parts in the other buffer. In this situation there is no routing decision
to be made, and \( H_0 \) may not hold a completed part while waiting for space to become available in the full buffer. When both buffers are full, the lead machine stops. It may not begin working on its next part, and it is said to be blocked. We assume that there is an unlimited supply or raw parts at the system input and an unlimited storage area at the system output. In other words, the lead machine is never starved and the downstream machines are never blocked. A machine which is either starved or blocked is said to be idle.

While machines may fail, we assume that buffers and transport mechanisms are perfectly reliable. Also, all transportation delays experienced by a part are assumed to be negligible in comparison with the processing times. Workpieces may wait in buffers, but the time they spend moving to, through, and from the buffers is ignored.

The order in which parts are selected from a buffer for processing at a downstream machine (i.e., the queueing discipline) is not specified in this model. First-in-first-out, last-in-first-out, or more complicated disciplines are possible. The only assumption is that the order in which parts are selected does not depend on their processing times.
2.1.4 Machines

The machines in this model are unreliable, with failures and repairs occurring randomly. A machine is always in one of two states: operational or under repair. An operational machine may actually be operating on a part, or it may be idle due to blockage or starvation. We assume that a machine is subject to failure only while it is operating on a part.

The periods of time between successive breakdowns of a machine, not including idle time or repair time, are modeled as independent, exponentially distributed random variables. The mean operating time between failures of machine $M_i$ is $\frac{1}{p_i}$, $i = 0, 1, 2$. If we consider only those time intervals when $M_i$ is actually operating, then breakdowns can be viewed as a Poisson process with average rate $p_i$, so we call $p_i$ the failure rate of machine $M_i$. The failure processes at different machines are assumed to be independent. This means, for instance, that the possibility of a power failure affecting several machines is not modeled.

Repair times are also assumed to be independent and exponentially distributed, with mean repair time for machine $M_i$ given by $\frac{1}{r_i}$, $i = 0, 1, 2$. If we consider only
those time intervals when $M_i$ is under repair, then
machine recoveries form a Poisson process with average
rate $r_i$, called the \textit{repair rate} of machine $M_i$. Repair
times at different machines are assumed to be indepen-
dent. In particular, this supposes that there is no
shortage of repair equipment or personnel; when more
than one machine is under repair, each machine can be
repaired as quickly as if it were the only failed
machine.

The time required for a machine to perform an
operation on a part, not including repair time, is
called the \textit{processing time} of that operation. Process-
ing times at machine $M_i$ are assumed to be independent
and exponentially distributed, with mean \textit{processing time}
given by $\frac{1}{\mu_i}$, $i=0,1,2$. If we consider only those time
intervals when $M_i$ is actually operating, then we can
describe part completions at that machine as a Poisson
process with average rate $\mu_i$, called the \textit{processing
rate} of machine $M_i$. As explained in section 2.1.2, the
processing times for all operations at all machines are
assumed to be independent, including the times required
for the two operations on the same part. Furthermore,
the processing times are independent of the repair times
and the times between failures at all the machines.
The various independence assumptions and exponential distributions described above permit us to construct a simple Markov model of a single unreliable machine in isolation. That is, we shall consider a machine which is removed from the manufacturing network and provided with an unlimited source of workpieces to process and an unlimited storage area for finished parts, so that the machine is never idle. As shown in figure 2.1, an isolated machine with processing, failure and repair rates of $\mu$, $p$ and $r$, respectively, can be modeled as a continuous-time Markov process with two states: "up" (operational) and "down" (under repair). In this figure, circles represent states, and links are marked with the corresponding transition rates. The self-transition from the "up" state (marked "$\mu$") corresponds to a part completion.

The exponential distributions for processing times, repair times and times between failures are assumed primarily to permit such simple, tractable, Markov models of the individual machines and of the larger network. (The network model is presented in sections 2.2.1 and 2.2.2.) These distributions are not unreasonable for failure and repair times in some applications (Schick and Gershwin [1978, pp. 34-36]). However, the exponen-
Figure 2.1. Transition diagram for an unreliable machine in isolation.
tial processing time assumption is unrealistic in most cases and, as such, is the weakest point of this model. (Following Gershwin and Berman [1978] and Berman [1979], we could have modeled processing times more realistically with Erlang distributions, but only at great cost in terms of the size and complexity of the system model.)

Using this concept of a machine in isolation, we can define several machine performance measures as functions of the basic parameters $\mu_i$, $p_i$ and $r_i$. The isolated efficiency $e_i$ of machine $M_i$ is the fraction of time the machine would be operational if it were in isolation. In other words, the isolated efficiency is the steady-state probability of the "up" state in figure 2.1. In terms of the machine parameters:

$$e_i = \frac{r_i}{r_i + p_i} \quad (2.1)$$

(Gershwin and Berman [1978, p.27], [1979, p. 10]).

The isolated production rate $\rho_i$ of machine $M_i$ is defined as the average rate at which the machine would produce parts in isolation, including time spent under repair. (This is in contrast to the processing rate $\mu_i$, which is measured by ignoring down time.) In terms of
the Markov machine model of figure 2.1, the isolated production rate is the steady-state average rate at which the self-transition for the "up" state occurs. Since the machine in isolation produces parts at rate $\mu_i$ when it is operational, and since $e_i$ is the fraction of time that this is the case, then:

$$\rho_i = \mu_i e_i = \frac{\mu_i r_i}{r_i + p_i}$$ (2.2)

(Gershwin and Berman [1978, p. 27], [1979, p. 10]).

The isolated efficiency $e_i$ and the isolated production rate $\rho_i$ are first-moment measures of the reliability of machine $M_i$, in that they measure the effect of breakdowns on the average rate of parts flow through the machine. Notice from (2.1) and (2.2) that $e_i$ and $\rho_i$ depend only on the ratio of $r_i$ to $p_i$. However, the magnitude of $r_i$ (in comparison to $\mu_i$) strongly affects the variability of the parts flow. Consider, for example, a machine with a low repair rate $r_i$ and an extremely low failure rate $p_i$. The machine rarely fails, but when it does, it is out of service for a long time. Such a machine is very reliable according to the first-moment measures $e_i$ and $\rho_i$: its average behavior in isolation is excellent. However, its parts flow is so variable that when the machine is placed in our manufacturing
network, the finite buffers cannot effectively smooth its output. Buffers are often empty or full, starvation and/or blockage occur over long time intervals, and system performance is poor. Such a machine is unreliable in a second-moment sense. It is shown precisely in appendix 2 how the failure and repair rates affect the mean, variance, and coefficient of variation of the time required for an unreliable machine to produce a part.

A manufacturing system is called **symmetric** if the two downstream machines are identical, that is, if \( \mu_1 = \mu_2, \rho_1 = \rho_2, \) and \( r_1 = r_2 \). A manufacturing system in which the capacity of the lead machine is matched to that of the downstream machines, that is, a system where \( \rho_0 = \rho_1 + \rho_2 \), is called a **balanced** system.

In some sections of this thesis, simplified system models will be used, in which some machines are assumed to be perfectly reliable. The part completion process is the same as that for an unreliable machine: processing times are independent and exponentially distributed with mean \( \frac{1}{\mu_1} \) at machine \( M_1 \). Notice that the output of a reliable lead machine during intervals when it is not blocked is a Poisson process. An equivalent model in this case, which is used frequently in applications other than manufacturing (as explained
in section 1.2), contains no lead machine but assumes instead that parts arrive in a Poisson stream at rate \( \mu_0 \), immediately enter one of the buffers, and are rejected if both buffers are full.

2.1.5 **System Parameters and System State**

In summary, a manufacturing network modeled as in the preceding sections can be described by the following system parameters:

- **Processing rates**: \( \mu_0, \mu_1, \) and \( \mu_2 \) (parts per unit time)
- **Failure rates**: \( p_0, p_1, \) and \( p_2 \) (failures per unit time)
- **Repair rates**: \( r_0, r_1, \) and \( r_2 \) (repairs per unit time)
- **Buffer capacities**: \( N_1 \) and \( N_2 \) (parts)

In simplified models where some machines are reliable, no failure and repair rates are specified at those machines.

We are also interested in a more detailed view of the network suitable for describing its dynamic behavior. Such a state space description includes the number of parts in each buffer and the operational status of each machine at any time. The state space is entirely discrete with a large number of possible system
states, which are usually indexed by i or j. Details of
the state space description are given in section 2.2.

2.1.6 Control Objective and Constraints

We are interested in finding a routing strategy
which will maximize the expected long-term production
rate of the manufacturing system. In terms of our sto-
chastic model, this quantity is the expected number of
parts completed at the downstream machines per unit time
when the system is in probabilistic steady state. Phy-
sically, this means that we intend to run our manufac-
turing system using the same routing strategy for a very
long time, long enough for the random disturbances to
eliminate the influence of the initial system state on
the behavior of the system. In measuring the production
rate, every completed part leaving the system has the
same value, and production is not discounted over time.

In the search for optimal routing strategies, we
restrict our attention to feedback control laws of the
following type. Routing decisions at time t may depend
on the state of the system at time t, but may not depend
on past system states, nor on past decisions, nor on
time t itself. Furthermore, we will not permit random-
ized strategies; the routing decision must be a deter-
ministic function of the system state. Feasible routing strategies will also be called policies.

A routing policy \( k \) or \( k(*) \), then, is simply a list whose \( i \)th element \( k(i) \) is the decision to be made whenever the system is in state \( i \). A decision can be either 1 or 2, indicating to which buffer a part should be routed. If one buffer is full, then \( k(i) \) must indicate the other buffer. If both buffers are full or the lead machine is under repair, then completion of a part by \( M_0 \) is impossible, so no routing decision is specified; we take \( k(i) \) to be 0 for such states.

The decision \( k(i) \) associated with state \( i \) is actually a tentative routing decision which will only be implemented if the next event to occur in the system is a part completion at \( M_0 \). If some other event occurs first (i.e., a part completion at a downstream machine, a failure, or a repair), then the system will move to a new state \( j \) and a new tentative decision \( k(j) \) will be selected.
2.2 **Formulation as a Continuous-Time Markov Decision Process**

The model described in section 2.1 can be formulated as a discrete-state, continuous-time Markov decision process. Elementary references on discrete-state Markov processes in general are Howard [1971] and Kleinrock [1975]. More advanced treatments can be found in Feller [1968], Feller [1971], Karlin [1966], Ross [1970], and Bharucha-Reid [1960]. Howard [1960], Derman [1970], and Wine and Osaki [1970] deal specifically with Markov decision processes. The primary references for this chapter are Howard [1960] and Howard [1971].

2.2.1 **State Space Description**

The state space description of our manufacturing system, like the rest of the model, is a direct extension of the transfer line work of Gershwin and Berman [1978, 1979]. The state space model requires at most five state variables, one for each machine and buffer. Buffer $B_1$ is described by its buffer level $n_1$. This specifies the number of parts in $B_1$, including the part in machine $M_1$, if any. (Specifically, if $n_1 = 0$, then both buffer $B_1$ and machine $M_1$ are empty. If $n_1 > 0$, then the machine contains one part, and the buffer con-
tains \( n_1 - 1 \) parts.) Buffer level \( n_1 \) can range from 0 to the buffer capacity \( N_1 \). Buffer level \( n_2 \) is defined similarly and ranges from 0 to \( N_2 \).

Each unreliable machine \( M_i \) is described by its machine state \( a_i \). When machine \( M_i \) is under repair, \( a_i = 0 \); when it is operational (either working on a part or idle due to starvation or blockage), \( a_i = 1 \). Reliable machines require no state variables.

The system state at time \( t \), for the complete model with all machines unreliable, is given by the 5-tuple \((n_1(t), n_2(t); a_0(t), a_1(t), a_2(t))\). For the simplified models with some reliable machines, the system state is denoted similarly, but with fewer than five dimensions. In some sections of this report, we will use this multivariable state notation. However, it will often be more convenient to index states by a single variable \( i \) or \( j \), running from 1 to \( S \).

The number of possible system states \( S \) can be very large, even for moderate buffer sizes, it all machines are assumed to be unreliable. The total number of states is given by:

\[
S = (N_1 + 1) \cdot (N_2 + 1) \cdot 2^c
\]  
(2.3)
where $c$ is the number of unreliable machines. For example, if $N_1 = N_2 = 24$ and all machines are unreliable, then there are 5000 system states.

It will be useful to depict the system state space as $2^c$ rectangular arrays of points, with one array for each combination of values of the unreliable machine state variables $a_i$. For example, one rectangle will represent that part of the state space where all machines are operational ($a_0 = a_1 = a_2 = 1$). Within an array, each point represents a pair of values of the buffer levels $n_1$ and $n_2$. We let $n_1$ vary along the $x$-axis of each plane and let $n_2$ vary along the $y$-axis. Figure 2.2 illustrates the state space for a system with a reliable lead machine, unreliable downstream machines, and buffer capacities $N_1 = 4$ and $N_2 = 3$. The edges of the rectangles are called the state space boundaries. In particular, those edges corresponding to $n_1 = N_1$ or $n_2 = N_2$ are called the upper state space boundaries.

Any routing strategy can be graphed by dividing each state space rectangle into areas called decision regions over which the routing decision is constant. The boundaries of the decision regions, called decision boundaries, completely characterize the routing policy. Of course, we need only consider those state space
Figure 2.2. State space for a system with a reliable lead machine.
planes where the lead machine is operational \((\sigma_0 = 1)\), since no decision is made when \(\sigma_0 = 0\). In other words, since the routing decision \(k = 0\) over all planes where \(\sigma_0 = 0\), we will not include these planes when graphing a policy. The decision \(k\) must also be 0 in the upper right-hand corner of every state space rectangle, since the lead machine is blocked when \(n_1 = N_1\) and \(n_2 = N_2\). Decisions are forced along the upper state space boundaries as well: \(k = 1\) if \(n_2 = N_2\), and \(k = 2\) if \(n_1 = N_1\). Except for these constraints, the decision regions may take any form within the rectangles. An example is shown in figure 2.3 for the state space of figure 2.2. The dashed lines are the decision boundaries, and the decision regions are marked with numerals indicating the decision \(k\). Since routing decisions are forced along the upper state space boundaries, we do not usually mention that part of the decision boundary when discussing the graph of a policy.

2.2.2 State Transitions

Our model is time-continuous; that is, transitions between system states may occur at any time. An event is defined as a single machine failure, repair, or part completion. Each system state transition corresponds to a single event in the network, since the probability of
\( \alpha_1 = 0 \quad \alpha_2 = 0 \)

\( \alpha_1 = 0 \quad \alpha_2 = 1 \)

\( \alpha_1 = 1 \quad \alpha_2 = 0 \)

\( \alpha_1 = 1 \quad \alpha_2 = 1 \)

---

Figure 2.3. A feasible routing policy.
two events happening at exactly the same instant is zero. The rate at which a system state transition occurs is given by the machine parameter $p_i$, $r_i$ or $\mu_i$ describing the frequency of the corresponding event.

Routing decisions affect only those transitions corresponding to a part completion at the lead machine. If the routing decision for state \( (n_1, n_2; a_0, a_1, a_2) \) is 1, then transitions from this state to state \( (n_1+1, n_2; a_0, a_1, a_2) \) occur at rate $\mu_0$. If the decision is 2, then transitions to state \( (n_1, n_2+1; a_0, a_1, a_2) \) occur at rate $\mu_0$. An example in table 2.1 shows all possible transitions from state \( (n_1, n_2; 1, 0, 1) \) and the rates at which they occur.

Once a routing policy $\mathcal{K}$ is specified, the possible state transitions and corresponding rates can be summarized in a transition rate matrix $A^\mathcal{K}$. The \((i,j)\)th element of $A^\mathcal{K}$ \((i \neq j)\) is equal to the transition rate from state $i$ to state $j$ if decision $k(i)$ of policy $\mathcal{K}$ is made in state $i$; this element is denoted $a^k_{ij}$. If a transition from $i$ to $j$ under policy $\mathcal{K}$ is impossible, then $a^k_{ij} = 0$. By convention, the diagonal elements $a_{ii}$ are defined so that the rows of $A^\mathcal{K}$ sum to zero:
<table>
<thead>
<tr>
<th>Event</th>
<th>Transition Rate</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part completion at $M_0$</td>
<td>$\mu_0$</td>
<td>[ \begin{cases} (n_1+1, n_2; 1, 0, 1) \ if decision is 1 \ (n_1, n_2+1; 1, 0, 1) \ if decision is 2 \end{cases} ]</td>
</tr>
<tr>
<td>Failure at $M_0$</td>
<td>$p_0$</td>
<td>$(n_1, n_2; 0, 0, 1)$</td>
</tr>
<tr>
<td>Repair at $M_1$</td>
<td>$r_1$</td>
<td>$(n_1, n_2; 1, 1, 1)$</td>
</tr>
<tr>
<td>Part completion at $M_2$</td>
<td>$\mu_2$</td>
<td>$(n_1, n_2-1; 1, 0, 1)$</td>
</tr>
<tr>
<td>Failure at $M_2$</td>
<td>$p_2$</td>
<td>$(n_1, n_2; 1, 0, 0)$</td>
</tr>
</tbody>
</table>

Table 2.1. Transitions from state $(n_1, n_2; 1, 0, 1)$, $0 < n_1 < N_1$, $0 < n_2 < N_2$. 
\[ a_{ii} = - \sum_{j=1}^{S} a_{ij}^{k(i)} \]

That is, \( a_{ii} \) equals the negative sum of the rates of all transitions out of state \( i \). (It happens that \( a_{ii} \) does not depend on the routing policy. This is because the decision \( k(i) \) determines only the destination state \( j \) to which the system moves when \( M_0 \) completes a part. The decision does not influence the rate \( \mu_0 \) at which that transition occurs.) If the states are indexed in a certain order, then the transition rate matrix will have a special structure which is discussed in detail in appendix 1.

Let us make some observations concerning the connectivity of the system states under any policy. From any system state \( (n_1, n_2; \sigma_0, \sigma_1, \sigma_2) \) it is possible to reach state \( (N_1, N_2; 1, 1, 1) \) by the following sequence of transitions:

1. (If \( \sigma_0 = 0 \)) \( M_0 \) is repaired.
2. (If \( \sigma_1 = 0 \)) \( M_1 \) is repaired.
3. (If \( \sigma_2 = 0 \)) \( M_2 \) is repaired.
4. \( M_0 \) completes \( N_1 + N_2 - n_1 - n_2 \) parts, routing them according to any policy.
It is also possible under any policy to go from state 
\((N_1, N_2; 1, 1, 1)\) to almost any state \((n_1, n_2; \sigma_0, \sigma_1, \sigma_2)\) (call 
this the target state) by a similar sequence of transi-
tions:

1. \(M_1\) completes \(N_1 - n_1\) parts.

2. \(M_2\) completes \(N_2 - n_2\) parts.

3. (If \(\sigma_0 = 0\)) \(M_0\) fails.

4. (If \(\sigma_1 = 0\)) \(M_1\) fails.

5. (If \(\sigma_2 = 0\)) \(M_2\) fails.

There are some states, however, which cannot be reached 
by this sequence of transitions. If \(\sigma_i\) of the target 
state is 0, but machine \(M_i\) is idle (either blocked or 
starved) after step 2 in the sequence, then the failure 
of machine \(M_i\) prescribed in step 3, 4 or 5 is impossi-
ble. The set \(P\) of states unreachable from \((N_1, N_2; 1, 1, 1)\) 
by this sequence is listed below:

\[(N_1, N_2; 0, \sigma_1, \sigma_2)\] for all \(\sigma_1, \sigma_2\)

\[(0, n_2; \sigma_0, 0, \sigma_2)\] for all \(n_2, \sigma_0, \sigma_2\)

\[(n_1, 0; \sigma_0, \sigma_1, 0)\] for all \(n_1, \sigma_0, \sigma_1\)

It is easy to see that these states cannot be reached
from \( (N_1, N_2, 1, 1, 1) \) by \textit{any} sequence of transitions, for the same reason: that idle machines cannot fail.

Let \( R \) denote the set of states not in \( P \). From the observations above, we conclude that any state in \( R \) can be reached from any other state in \( R \), regardless of the routing policy. The set \( R \) is called a recurrent chain, and states in \( R \) are called \textbf{recurrent}. We may also conclude from these observations that if the system is started in some state in set \( P \), it will eventually leave \( P \), and once outside of \( P \), the system can never return to \( P \) regardless of the routing policy. For this reason, the states in \( P \) are said to be \textbf{transient}. The Markov process is called \textbf{unichain} (or monodesmic or completely ergodic) because it has only one recurrent chain. (See Howard [1971, pp. 15, 31] for a discussion of these concepts.)

Since we are only interested in the steady state behavior of the manufacturing system, hereinafter we shall ignore the transient states and consider a model containing only the recurrent states. Such a unichain process with no transient states is said to be \textbf{irreducible}. The number or system states will still be denoted by \( S \), though equation (2.3) will no longer give an accurate count.
For simplified models with reliable machines, the list of transient states is altered in the obvious way.

2.2.3 Steady-State Probabilities

Assume that the routing policy $K$ is known and fixed and that we are given the probability that the system is in any state $i$ at time $t = 0$. Then let $\pi^K_i(t)$ denote the conditional probability that the system is in state $i$ at time $t > 0$. Because the Markov process model of the system is irreducible under any policy, $\pi^K_i(t)$ approaches a limit $\pi^K_i$ as $t$ approaches infinity, and this limit does not depend on the initial state probabilities (Howard [1971, p. 776], Kleinrock [1975, p. 52], Feller [1971, pp. 491-492]). The vector $\pi^K$ with components $\pi^K_i$ is called the limiting state probability vector or the steady-state probability vector. It can be shown that this vector is the unique solution of the equations:

$$ (\pi^K)' A^K = 0' $$  \hspace{1cm} (2.5)

$$ \sum_{i=1}^{S} \pi^K_i = 1 $$  \hspace{1cm} (2.6)

(Howard [1971, p. 777], Kleinrock [1975, p. 52]). It is evident from these equations and equation (2.4) that the transition rates $a^{k(i)}_{ij}$ can be scaled by an arbitrary multiplicative constant without changing the steady
state probabilities. This is equivalent to changing the
time unit in which the transition rates are measured.

2.2.4 Rewards

Since we are interested in maximizing the production rate of the system, let us define a transition reward $b_{ij}$ which equals 1 if a transition from state $i$ to state $j$ involves a part completion at a downstream machine and which equals 0 otherwise. Also define a reward rate $q_i$, which is the average rate at which parts are produced whenever the system is in state $i$. The reward rate can be expressed in terms of the transition rates and transition rewards as shown below:

$$q_i = \sum_{j=1}^{S} a_{ij} b_{ij}$$  \hspace{1cm} (2.7)

Alternatively, we can use the multivariable state notation, and express $q(n_1, n_2; a_0, a_1, a_2)$ as a function of the system parameters and state variables:

$$q(n_1, n_2; a_0, a_1, a_2) = \mu_1 \cdot \{a_1=1\} \cdot \{n_1>0\}$$  \hspace{1cm} (2.8)

$$+ \mu_2 \cdot \{a_2=1\} \cdot \{n_2>0\}$$

Each set of braces $\{\}$ above represents an indicator function which equals 1 if the expression within the
braces is true and which equals 0 otherwise. It is clear from (2.8) that \( q_i \) does not depend on the routing policy \( K \). Finally, we define a vector \( q \) whose components are the reward rates \( q_i \).

2.2.5 **Optimization Objective**

We are now able to formulate an expression for the production rate \( g^K \) of the manufacturing system governed by routing policy \( K \). Since \( \pi^K_i \) is the steady-state probability that the system is in state \( i \), and since the system produces parts at the reward rate \( q_i \) when it is in that state, we define \( g^K \) as follows:

\[
g^K = \sum_{i=1}^{S} \pi^K_i q_i = (\pi^K)' q \quad (2.9)
\]

Howard [1960, p. 103] proves that if the system is started in any state \( i \) and runs for a long time, then the limiting time-average of the expected number of parts produced is given by (2.9). Our objective is to find a policy \( K \) which maximizes \( g^K \). Recall that the policy \( K \) determines the transition rate matrix \( \Lambda^K \), which in turn determines \( \pi^K \) through equations (2.5) and (2.6), which then determines \( g^K \) by equation (2.9).
2.3 An Equivalent Discrete-Time Model

The algorithm used in this thesis to determine optimal routing strategies requires a discrete-time Markov process model. It is therefore necessary to adapt the continuous-time model presented in sections 2.1 and 2.2. Let us measure the transition rates of the continuous-time model using a small time unit, so that the quantities $a_{ij}^{k(i)}$ are small compared to 1. Let us take the same time unit as the fundamental time step of a discrete-time Markov process with the same state space as the continuous-time model and with identical transition and decision structures. The discrete-time model is also unichain, and it has the same transient states as the original model; these transient states will also be ignored. A transition from state $i$ to state $j$ ($i \neq j$), which occurs at a rate of $a_{ij}^{k(i)}$ transitions per unit time in the continuous-time model, is defined to occur with probability $t_{ij}^{k(i)}$ in one time step of the discrete-time model, where:

$$t_{ij}^{k(i)} = a_{ij}^{k(i)} \text{ for } i \neq j \quad (2.10)$$

The probability $t_{ii}$ of no transitions from state $i$ in one time step is necessarily given by:
\[ t_{ii} = 1 - \sum_{\substack{j=1 \atop j \neq i}}^{S} t_{ij} \]
\[ = 1 - \sum_{\substack{j=1 \atop j \neq i}}^{S} a_{ij} \]
\[ = 1 + a_{ii} \]

making use of equation (2.4). Like \( a_{ii} \), \( t_{ii} \) does not depend on the routing policy \( K \). (See section 2.2.2.) It is convenient to consider \( t_{ii} \) as the probability of a transition from state \( i \) to itself, called a self-transition. By construction, then, the transition probability matrix \( T^K \) with elements \( t_{ij}^K \) is related to the transition rate matrix \( \lambda^K \) as follows:

\[ T^K = I + \lambda^K \]  

where \( I \) is the identity matrix.

In order for the discrete-time model to be meaningful, the time unit must be small enough so that \( t_{ii} > 0 \). From (2.11) and (2.12), this means that:

\[ \sum_{\substack{j=1 \atop j \neq i}}^{S} t_{ij}^K = \sum_{\substack{j=1 \atop j \neq i}}^{S} a_{ij} < 1 \quad \text{for all } i \]  

This condition is sufficient to guarantee another
important condition, namely that:

\[ t_{ij} = a_{ij} < 1 \quad \text{for all } i, j, k(i) \quad (2.16) \]

Let the discrete time steps be indexed by \( m \). Then the state probabilities \( \pi^k_i(m) \) for the discrete-time model are defined analogously to \( \pi^k_i(t) \) in the continuous-time model. The existence of limiting (steady-state) probabilities \( \pi^k_i \), as \( m \) approaches infinity, is guaranteed because the new model is irreducible and contains self-transitions. As before, the limiting probabilities do not depend on the initial state probabilities. (See Kleinrock [1975, p. 30] and Feller [1968, p. 393], and note that the self-transitions guarantee aperiodicity.) Again we define a steady-state probability vector \( \pi^K \). It can be shown that this vector is the unique solution of the equations:

\[
\begin{align*}
\left(\pi^K\right)' \pi^K &= \left(\pi^K\right)' \\
\sum_{i=1}^{S} \pi^K_i &= 1
\end{align*}
\]

(Howard [1971, p. 16]).

This new model could have been derived as an approximation to the continuous-time model by discretiz-
ing time into extremely small steps. The time step would have to be so small that the exponentially distributed processing times, times between failures, and repair times of the original model could be closely approximated in the discrete-time model by geometric distributions with the same means. The time step would also have to be small enough that the probability of two events occurring in a single time step would be negligible. We see, then, that the continuous-time and discrete-time models match only in the limit as the time unit approaches zero. Therefore one would guess that the steady-state probabilities of the two models would be the same only in the limit as the time unit and hence the quantities $a_{ij}^{k(i)}$ approach 0. However, a comparison of equations (2.5), (2.6), (2.14), (2.17), and (2.18) shows that the two models have exactly the same steady state probabilities for any time unit small enough so that the $a_{ij}^{k(i)}$ satisfy condition (2.15). Since

$$\pi_i^K = \pi_i^K$$

for all $i, K$ (2.19)

hereinafter we shall use the same symbol $\pi_i^K$ for both models.

The reward structure for the new model will be very similar to that defined in section 2.2.4 for the origi-
nal model. Transition rewards $b_{ij}$ are identical. We define the expected immediate reward $\theta_i$ as the expected number of parts to be produced in the next time step, given that the system is currently in state $i$; that is:

$$\theta_i = \sum_{j=1}^{S} t_{ij} b_{ij} \quad (2.20)$$

Since a transition occurs every time step, and since this time step matches the time unit of the continuous-time model, then the expected immediate reward $\theta_i$ of the discrete-time model is analogous to the reward rate $q_i$ of the continuous-time model. In fact, we see from equations (2.7), (2.10) and (2.20) that $q_i$ and $\theta_i$ are identical, so hereinafter we shall use the same symbol $q_i$ for both. The state variable expression (2.8) for $q_i$ is also valid for both the discrete-time and continuous-time models.

Since $\pi_i^K$ is the steady-state probability that the discrete-time system is in state $i$, and since the system produces an expected number $q_i$ of parts per time step (unit time) whenever it is in state $i$, we define the production rate $\theta_i^K$ of the discrete-time system under policy $K$ as follows:
\[ g^K = \sum_{i=1}^{S} \pi^K q_i = (\pi^K)' q \quad (2.21) \]

Howard [1960, p. 23] proves that if the system is started in any state \( i \) and runs for many transitions, then the limiting time-average of the expected number of parts produced is given by (2.21). Notice that this formula is the same as equation (2.9) for the production rate \( g^K \) of the continuous-time model; hereinafter we shall use the same symbol \( g^K \) for the production rate of either model under policy \( K \). Since any policy applied to both models will yield the same production rate, then a policy which is optimal for one model will be optimal for the other. (Howard [1960, pp. 112-114] shows this in a different way.) Thus the discrete-time model can be used to compute optimal routing strategies for the original model without introducing error.
2.4 *Howard's Dynamic Programming Formulation*

Our objective is to find a routing policy $K$ which maximizes the steady state production rate $g^K$ of the manufacturing system. One possibility is to compute $g^K$ for every feasible policy $K$, since there are a finite number of possible policies. This is not practical, however, because the number of possible policies is enormous. (Since a decision between two alternatives is required for every state where the lead machine is capable of producing a part and where space is available in both buffers, and since there are approximately $4N_1N_2$ such states, then the number of possible policies is roughly $2^{4N_1N_2}$ or $16N_1N_2$.) Instead we can take a dynamic programming approach which is much more efficient. This section briefly summarizes Howard's dynamic programming formulation or the decision problem for discrete-time, unichain Markov processes (Howard [1960]).

Define $v_i^*(m)$ to be the expected number of parts produced by the system in the next $m$ time steps if the system is now in state $i$ and an optimal policy is used. By the "Principle of Optimality" of dynamic programming (Bellman [1972, p. 83]), the $v_i^*(m)$ satisfy the following set of equations (Howard [1960, p. 29]):
\[ v_i^{*}(m+1) = \max_{k(i)} \left[ \sum_{j=1}^{S} t_{ij} k(i) \left( b_{ij} + v_j^{*}(m) \right) \right] \] (2.22)

\[ = q_i + \max_{k(i)} \left[ \sum_{j=1}^{S} t_{ij} v_j^{*}(m) \right] \] (2.23)

for all \( i \), for all \( m \geq 0 \)

The first term on the right side of (2.23) is the expected immediate reward during the next time step. The second term is the expected future reward over all time steps subsequent to the first. Note that the maximization is over all possible decisions \( k(i) \) which can be made in state \( i \), not over all possible policies.

Howard shows that as \( m \) increases, \( v_i^{*}(m) \) asymptotically approaches a linear function of \( m \):

\[ v_i^{*}(m) \longrightarrow v_i^{*} + mg^{*} \quad \text{for all } i \] (2.24)

That is,

\[ \lim_{m \to \infty} \left[ v_i^{*}(m) - (v_i^{*} + mg^{*}) \right] = 0 \quad \text{for all } i \] (2.25)

Here \( g^{*} \) is the optimal production rate. The asymptotic intercept \( v_i^{*} \) will be called the value of state \( i \). We define a value vector \( \mathbf{v}^{*} \) with components \( v_i^{*} \).

Substituting the asymptotic expression (2.24) for \( v_i^{*}(m) \) into equation (2.23) and simplifying yields
Howard's equations:

\[ v_i^* + g^* = q_i + \max_{k(i)} \left[ \sum_{j=1}^{S} t_{ij}^k v_j^* \right] \text{ for all } i \] (2.26)

In these equations, \( q_i \) and \( t_{ij}^k \) are known, while \( v_i^* \) and \( g^* \) are unknown. Schweitzer [1965, p. 149] shows that Howard's equations (in our case) always have a solution with \( g^* \) unique and \( v_i^* \) unique up to a single additive constant. The optimal policy may not be unique; any value of \( k(i) \) which yields the maximum in the \( i \)th Howard equation for any solution \((g^*, v^*)\) of these equations will be an optimal decision for state \( i \).

A more detailed version of Howard's equations can be written in terms of the system parameters and state variables of the model. Since most or the transition probabilities \( t_{ij}^k \) are zero, it is possible to write out all the terms of equation (2.26). Moreover, for any state \( i \) there are very few transition probabilities \( t_{ij}^k \) which actually depend on the routing decision \( k(i) \). If state \( i \) includes a failed lead machine or one or two full buffers, then none of the transition probabilities depend on \( k(i) \). If state \( i \) includes an operational lead machine and buffers with available space, then only two transition probabilities depend on \( k(i) \); these correspond to adding a part to buffer \( B_1 \) and
adding a part to buffer \( B_2 \). One of these transition probabilities must equal \( \mu_0 \) and the other must equal \( 0 \); the routing decision \( k(i) \) specifies which is which. Thus the maximization in equation (2.26) reduces to a simple comparison of the values \( v^* \) of two states \( j \). Specifically, to perform the maximization for state \( i \) (where this state includes an operational lead machine and buffers with available space), we compare the value of the state \( i' \) attained by placing the next part in \( B_1 \) with the value of the state \( i'' \) attained by placing the next part in \( B_2 \), and we select the routing decision \( k(i) \) corresponding to the larger quantity. The resulting detailed version of Howard's equations is given below.
\[ v^*(n_1, n_2; a_0, a_1, a_2) + g^* \] (2.27)

\[ = \mu_1 \cdot \{ a_1 = 1 \} \cdot \{ n_1 > 0 \} + \mu_2 \cdot \{ a_2 = 1 \} \cdot \{ n_2 > 0 \} + \mu_0 \cdot \{ a_0 = 1 \} \cdot \{ n_1 = N_1 \} \cdot \{ n_2 < N_2 \} \cdot v^*(n_1, n_2 + 1; a_0, a_1, a_2) + \{ n_1 < N_1 \} \cdot \{ n_2 = N_2 \} \cdot v^*(n_1 + 1, n_2; a_0, a_1, a_2) + \{ n_1 < N_1 \} \cdot \{ n_2 < N_2 \} \cdot \max (v^*(n_1 + 1, n_2; a_0, a_1, a_2), v^*(n_1, n_2 + 1; a_0, a_1, a_2)) \]

\[ + p_0 \cdot \{ a_0 = 1 \} \cdot \{ n_1 > N_1 \} \cup n_2 < N_2 \} \cdot v^*(n_1, n_2; 0, a_1, a_2) + \tau_0 \cdot \{ a_0 = 0 \} \cdot v^*(n_1, n_2; 1, a_1, a_2) \]

\[ + \mu_1 \cdot \{ a_1 = 1 \} \cdot \{ n_1 > 0 \} \cdot v^*(n_1 - 1, n_2; a_0, a_1, a_2) + \mu_1 \cdot \{ a_1 = 1 \} \cdot \{ n_1 > 0 \} \cdot v^*(n_1, n_2; a_0, 0, a_2) + \tau_1 \cdot \{ a_1 = 0 \} \cdot v^*(n_1, n_2; a_0, 1, a_2) \]

\[ + \mu_2 \cdot \{ a_2 = 1 \} \cdot \{ n_2 > 0 \} \cdot v^*(n_1, n_2 - 1; a_0, a_1, a_2) + \mu_2 \cdot \{ a_2 = 1 \} \cdot \{ n_2 > 0 \} \cdot v^*(n_1, n_2; a_0, a_1, 0) + \tau_2 \cdot \{ a_2 = 0 \} \cdot v^*(n_1, n_2; a_0, a_1, 1) \]

\[ + [1 - (\mu_0 + p_0) \cdot \{ a_0 = 1 \} \cdot \{ n_1 < N_1 \} \cup n_2 < N_2 \} - \tau_0 \cdot \{ a_0 = 0 \} + (\mu_1 + p_1) \cdot \{ a_1 = 1 \} \cdot \{ n_1 > 0 \} - \tau_1 \cdot \{ a_1 = 0 \} + (\mu_2 + p_2) \cdot \{ a_2 = 1 \} \cdot \{ n_2 > 0 \} - \tau_2 \cdot \{ a_2 = 0 \} ] \cdot v^*(n_1, n_2; a_0, a_1, a_2) \]
Each set of braces {} in the equation above represents an indicator function which equals 1 if the expression within the braces is true and which equals 0 otherwise. Due to the length and complexity of equations (2.27), we shall avoid the multivariable state notation and deal instead with equations (2.26) in the next chapter, where we discuss the algorithm used to solve Hovard's equations.
2.5 Summary

In this chapter we present a discrete-state, continuous-time, Markov decision model for our manufacturing system. Internal storage buffers have limited capacity. Machines are unreliable, with exponentially distributed repair times and times between failures. Processing times at each machine are also exponentially distributed. Different machines may have different processing, failure and repair rates, and the buffers may have different capacities. The system state includes the operational status of each machine and the number of parts in each buffer, resulting in a very large state space.

The problem of finding a dynamic routing policy to maximize the steady-state production rate of the system is easily stated in terms of the model, but to render the problem solvable by existing algorithms, two changes must be performed. First, a discrete-time model which is exactly equivalent for our purposes is constructed. Second, the optimization is restated using Howard's dynamic programming formulation. In the next chapter, we present the dynamic programming algorithm used in this thesis for computing optimal routing strategies for the discrete-time model.
3. DESCRIPTION OF THE ALGORITHM

In this chapter we describe the iterative algorithm used to compute optimal routing policies. The basis of the algorithm is D. J. White's method of successive approximations (White [1963]) for solving Howard's equations (2.26). This is presented in section 3.1. Various modifications to White's method are introduced in sections 3.2, 3.3 and 3.4 to improve its rate of convergence, and the version actually implemented is described in section 3.5. The convergence criterion is discussed in section 3.6, and a few remarks concerning the computational complexity of the algorithm appear in section 3.7.

This chapter deals almost exclusively with the discrete-time model of chapter 2. The few references to the continuous-time model will be clearly indicated as such.

3.1 White's Method of Successive Approximations

In this section we summarize D. J. White's method of successive approximations (White [1963]) for solving Howard's equations (2.26). For convenience, we restate these equations below:
\[ v^*_i + g^* = q_i + \max_{k(i)} \left[ \sum_{j=1}^{S} t_{ij} v^*_j \right] \quad \text{for all } i \quad (3.1) \]

Since these equations determine the values \( v^*_i \) only up to a single additive constant, and since any solution of these equations will give the correct policy and optimal production rate \( g^* \), we can set the value of any single state arbitrarily. We take \( v^*_1 \) to be zero. The values \( v^*_i \) will now be called the \textit{relative values} of the states. The relative values have a simple physical interpretation: \( v^*_i \) is the expected difference in long-run total production if the system is started in state \( i \) rather than in state 1. Howard's equations now become:

For \( i=1 \): \[ g^* = q_1 + \max_{k(1)} \left[ \sum_{j=2}^{S} t_{1j} v^*_j \right] \quad (3.2) \]

For \( i>1 \): \[ v^*_i = -g^* + q_i + \max_{k(i)} \left[ \sum_{j=2}^{S} t_{ij} v^*_j \right] \quad (3.3) \]

Equations (3.2) and (3.3) can be used as the basis for an iterative scheme to compute \( g^* \) and \( v^*_i \). Let \( m \) denote the iteration number, and let \( g(m) \) and \( v^*_i(m) \) denote the estimates at the \( m \)-th iteration of \( g^* \) and \( v^*_i \), respectively. Initial estimates \( g(0) \) and \( v^*_i(0) \) can be set arbitrarily; we take them to be zero. At each
iteration, first the production rate is updated according to equation (3.2), using the relative values from the previous iteration:

\[ c(m+1) = q_1 + \max_{k(1)} \left[ \sum_{j=2}^{S} t_{1j}^{k(1)} v_j(m) \right] \] (3.4)

Then the relative values are updated according to equation (3.3), using the new estimate of the production rate and the old estimates of the relative values:

\[ v_i(m+1) = -c(m+1) + q_i + \max_{k(i)} \left[ \sum_{j=2}^{S} t_{ij}^{k(i)} v_j(m) \right] \] (3.5)

for \( i > 1 \)

Equations (3.4) and (3.5) are called White's equations.

White showed that this iterative scheme converges to the solution of Howard's equations under certain conditions which are difficult to verify (White [1963]). Schweitzer proved that the method converges as long as the Markov process is unichain and aperiodic under any policy (Schweitzer [1965]), p. 290). In chapter 2 we showed that our Markov model satisfies these conditions, so convergence of White's algorithm is guaranteed for our problem.
White's equations can also be used to compute the production rate of a given suboptimal strategy. Rather than performing the maximization over all feasible decisions \( k(i) \), however, we simply use the decision specified by the suboptimal policy. This technique is used to determine the performance of various heuristic routing strategies for comparison with the optimal policy in chapter 6.

3.2 Time Scale Selection

Recall that we have some freedom in choosing the time scale in which the transition rates \( a_{ij}^{k(i)} \) and the production rate \( g^K \) of the continuous-time model are measured. This choice fixes the time unit of the equivalent discrete-time model. As noted in section 2.3, the only restriction is that:

\[
t_{ii} = 1 - \sum_{j=1}^{S} t_{ij}^{k(i)} > 0 \quad \text{for all } i \quad (3.6)
\]

Experience with the application of White's algorithm to our manufacturing model shows that the convergence rate improves dramatically if the time unit is selected as large as possible subject to (3.6). The computer program written for this thesis incorporates a modification
to White's equations (described below) which has the same effect as scaling the time unit so that:

\[
\min_{i} t_{ii} = 1 - \max_{\substack{i \neq 1 \not\in i}} \sum_{j=1}^{S} t_{ij}^{k(i)} = 0.01 \quad (3.7)
\]

(According to Stewart [1978], Wallace and Rosenberg [1966] found the same beneficial effect of lengthening the time unit when iteratively computing the steady-state probabilities of Markov models.)

Let us rewrite White's equations to incorporate an effective time scaling by a factor \( \theta \). (To lengthen the time unit, take \( \theta > 1 \).) If \( q_{i} \) and \( t_{ij}^{k(i)} \) are in the original time units, and it is desired to compute \( g^{*} \) in the original time units, then simply make the following substitutions in equations (3.4) and (3.5):

1. Replace \( g \) by \( \theta g \).
2. Replace \( q_{i} \) by \( \theta q_{i} \).
3. Replace \( t_{ij}^{k(i)} \) by \( \theta t_{ij}^{k(i)} \) for \( i \neq j \).
4. Replace \( t_{ii} = 1 - \sum_{\substack{j=1 \not\in i}}^{S} t_{ij}^{k(i)} \) by \( 1 - \sum_{\substack{j=1 \not\in i}}^{S} \theta t_{ij}^{k(i)} \).
The relative values $v_i$ are not changed in the equations, since they do not depend on the time unit. White's equations become:

$$\theta g(m+1) = \theta q_1 + \max_{k(1)} \left[ \sum_{j=2}^{S} \theta t_{ij} v_j(m) \right] \quad (3.8)$$

$$v_i(m+1) = -\theta g(m+1) + \theta q_i$$

$$+ \max_{k(i)} \left[ \sum_{j=2}^{S} \theta t_{ij} v_j(m) \right]$$

$$+ \left[ 1 - \sum_{j=1}^{S} \theta t_{ij} v_i(m) \right] v_i(m) \quad \text{for } i > 1$$

Cancelling $\theta$ from each term of (3.8), we recover the original equation (3.4). Equation (3.9) can be simplified by adding $(1 - \theta) v_i(m)$ outside the brackets and subtracting it from within the brackets:
\[ v_i(m+1) = (1 - \theta) v_i(m) - \theta g(m+1) + \theta q_i \quad (3.10) \]

\[ + \max_{k(i)} \left[ \sum_{j=2}^{\infty} \theta t_{ij} k(i) v_j(m) \right] \]

\[ + \left[ \theta - \sum_{j=1}^{\infty} \theta t_{ij} \right] v_i(m) \quad \text{for } i > 1 \]

Using the defining equation (2.11) for \( t_{ii} \), this becomes:

\[ v_i(m+1) = (1 - \theta) v_i(m) - \theta g(m+1) + \theta q_i \quad (3.11) \]

\[ + \max_{k(i)} \left[ \sum_{j=2}^{\infty} \theta t_{ij} k(i) v_j(m) \right] + \theta t_{ii} v_i(m) \]

\[ + \theta t_{ii} v_i(m) \quad \text{for } i > 1 \]

\[ v_i(m+1) = (1 - \theta) v_i(m) - \theta g(m+1) + \theta q_i \quad (3.12) \]

\[ + \theta \max_{k(i)} \left[ \sum_{j=2}^{\infty} t_{ij} k(i) v_j(m) \right] \quad \text{for } i > 1 \]

Equations (3.4) and (3.12) summarize White's method with
3.3 Acceleration Procedure

Kushner and Kleinman [1971] recommend the following technique to speed convergence of iterative solutions for Markov control problems similar to ours. They call this the "acceleration" procedure; it corresponds to a standard technique called "overrelaxation" for iterative solution of systems of simultaneous linear equations. (See, for example, Varga [1962] and Young [1971].)

If \( \mathbf{v}(m) \) is the vector of approximate relative values \( v_i(m) \) at the \( m \)th iteration of White's algorithm and \( \mathbf{v}(m+1) \) is the \( (m+1) \)st iterate, then we replace \( \mathbf{v}(m+1) \) by

\[
\mathbf{v}(m+1) = \mathbf{v}(m) + \omega \cdot (\mathbf{v}(m+1) - \mathbf{v}(m)) \quad (3.13)
\]

The scalar factor \( \omega \) is called the acceleration parameter and is usually taken between 1 and 2. The acceleration procedure is simply a linear extrapolation at each iteration by this fixed amount.

Incorporating this procedure, White's equation (3.5) becomes:
\[ \nu_i(m+1) = (1 - \omega) \nu_i(m) \]
\[
+ \omega \left[ -q_i(m+1) + q_i + \max_{k(i)} \left[ \sum_{j=2}^{S} t_{ij} k(i) \nu_j(m) \right] \right]
\]

for \( i > 1 \)

White's equation (3.4) is unchanged. Surprisingly, equation (3.14) is identical to equation (3.12) of the preceding section, if we identify \( \omega \) with \( \delta \). We conclude that the acceleration procedure is equivalent to lengthening the time unit of the model by an additional factor \( \omega \). The procedure is guaranteed to converge for \( \omega \) between 0 and 1, since it is simply White's method with a shortened time unit. But if the system parameters have already been scaled to satisfy equation (3.7), then a choice of \( \omega \) greater than about 1.01 will violate condition (3.6), and convergence is no longer guaranteed. However, in the author's experience with this particular manufacturing problem, the algorithm converges even when \( \omega \) is so large that some of the scaled probabilities \( t_{ii} \) are negative. Generally, the convergence rate increases as \( \omega \) increases, up to a limit. As \( \omega \) moves beyond that limit, the convergence rate decreases, and then, for sufficiently large \( \omega \), the method diverges. The region of convergence and the best value of \( \omega \) must be found by
trial and error. In the computer program used for this thesis, first one time scale factor $\theta$ is computed to satisfy equation (3.7). This brings the scaled probabilities close to the values which maximize the convergence rate. For fine tuning, an additional factor $\omega$ is determined interactively by trial and error as the algorithm proceeds. Convergence is usually fastest for some value of $\omega$ between 1.2 and 1.6.

3.4 **Successive Substitution Procedure**

Another technique recommended by Kushner and Kleinman [1971], called "successive substitution," has been implemented to improve the convergence rate of the algorithm. A second advantage of this procedure is that it cuts computer memory requirements in half. Kushner and Kleinman also call this the "Gauss-Seidel" technique after a similar method for iterative solution of systems of simultaneous linear equations. (See, for example, Varga [1962] and Young [1971].)

In the Gauss-Seidel version of White's algorithm, rather than computing each new relative value $v_{i}(m+1)$ as a function of all the old relative values $v_{j}(m)$, instead we use any new relative values $v_{j}(m+1)$ that have already been computed. White's equation (3.5) becomes:
\[ v_i(m+1) = -g(m+1) + q_i \tag{3.15} \]

\[ + \max_{k(i)} \left[ \sum_{j=2}^{i-1} t_{ij} k(i) v_j(m+1) \right] + \left[ \sum_{j=i}^{S} t_{ij} k(i) v_j(m) \right] \]

for \( i > 1 \)

White's equation (3.4) is unchanged. Though an improvement in the rate of convergence cannot be guaranteed (in fact, convergence itself is not guaranteed for the altered algorithm), in practice we find that the Gauss-Seidel procedure is beneficial. (For problems with certain special structures, it can be proven that the procedure is an improvement. See Varga [1962], Young [1971], and Kushner and Kleinman [1968] for examples. Our problem, however, does not fall into these special categories.)

3.5 Algorithm Actually Implemented

Combining the changes to White's algorithm introduced in the last few sections (time scaling, acceleration and successive substitution), we arrive at the iterative procedure actually used in this study for computing optimal routing policies.
Initialization:

\[ g(0) = 0 \] (3.16)

\[ v_i(0) = 0 \quad \text{for } i > 1 \] (3.17)

Iteration:

\[ g(m+1) = q_1 + \max_{k(1)} \left[ \sum_{j=2}^{S} t_{1j}^k v_j(m) \right] \] (3.18)

\[ v_i(m+1) = (1 - \omega) v_i(m) \]

\[ + \omega \left[ -g(m+1) + q_1 \right. \]

\[ + \max_{k(i)} \left[ \sum_{j=2}^{i-1} t_{ij}^k v_j(m+1) \right] + \left[ \sum_{j=i}^{S} t_{ij}^k v_j(m) \right] \]

\[ \text{for } i > 1 \]

The combination of acceleration and successive substitution is called "accelerated Gauss-Seidel" by Kushner and Kleinman [1971]. In the literature on numerical methods for linear equations, it is called "successive overrelaxation (SOR)" (Varga [1962], Young [1971]).

As mentioned in section 3.3, \( \theta \) is computed once at
the beginning of the algorithm so that the scaled transition probabilities satisfy equation (3.7). That is, \( \theta \) must satisfy:

\[
1 - \max_i \sum_{j=1}^{S} \theta_{ij}^k = 0.01 \quad (3.20)
\]

This is equivalent to:

\[
\theta = \frac{0.99}{\max_i \sum_{j=1}^{S} \theta_{ij}^k} \quad (3.21)
\]

It is simple to evaluate this maximum in terms of the machine parameters, since the number of possible transitions out of any state is limited. For a system model with all machines unreliable:

\[
\theta = \frac{0.99}{\max \left[ (\mu_0 + P_0), r_0 \right] + \max \left[ (\mu_1 + P_1), r_1 \right] + \max \left[ (\mu_2 + P_2), r_2 \right]} \quad (3.22)
\]

This initial scaling is done to move the transition probabilities close to the values which maximize the convergence rate. Fine tuning is done interactively by
varying $\omega$ as the algorithm proceeds.

The amount of computation involved in equations (3.18) and (3.19) is much less than it appears. The reasons for this were brought out in section 2.4 when Howard's equations (2.26) were rewritten in multivariable state notation (equation 2.27). First, almost all of the transition probabilities $t_{ij}^{k(i)}$ are zero; from each state $i$ there are no more than seven possible transitions, including the self-transition. Second, the maximization specified in equations (3.18) and (3.19) reduces, at worst, to a simple comparison of the relative values of two states $j$. That is, for a state $i$ where the lead machine is capable of producing a part, we compare the relative value of the state $i'$ reached by placing the next part in buffer $B_1$ with the relative value of the state $i''$ reached by placing the next part in buffer $B_2$, and we make the routing decision $k(i)$ corresponding to the larger quantity. Third, we can organize the computations to take advantage of these features by rewriting equations (3.18) and (3.19) using the multivariable state notation of section 2.2.1 and programming in nested loops, one for each state variable. (See the multivariable version of Howard's equations (2.27).) The algorithm is much easier to program.
in a language with an "if-then-else" statement, such as PL-1. This statement is extremely convenient for testing various conditions of the state variable values (specifically, the conditions in braces in equation (2.27)) in order to determine the nonzero transition probabilities \( t_{ij}^k(i) \) for each state \( i \).

3.6 Convergence Criterion

It is important to know when the relative value approximations \( v_i(m) \) computed by our iterative procedure are close enough to the true relative values \( v_i^* \) so that the latest routing policy found by the algorithm is guaranteed to be the optimal policy. As discussed in section 3.5, the relative values must be sufficiently resolved that the following comparisons can be made correctly: for each state \( i \) where the lead machine \( M_0 \) is capable of producing a part, the relative value of the state \( i' \) achieved by placing another part in buffer \( B_1 \) must be compared with the relative value of the state \( i'' \) achieved by placing another part in \( B_2 \). To estimate the degree of resolution required, we can compute a critical difference \( d(m) \) at each iteration \( m \):
\[ d(m) = \min_{\text{states } i \text{ where } M_0 \text{ can produce a part}} |v_{i^*}(m) - v_i^*(m)| \quad (3.23) \]

Let us define the maximum error \( e(m) \) in the approximate relative values as follows:

\[ e(m) = \max_i |v_i(m) - v_i^*| \quad (3.24) \]

If the maximum error \( e(m) \) is less than \( \frac{1}{2} d(m) \) at any iteration \( m \), then we can be certain that the algorithm has found the optimal policy. To see this, consider the case where:

\[ v_{i^*}(m) > v_i^*(m) \quad (3.25) \]

so that the decision for state \( i \), given by the algorithm at iteration \( m \), is to route the next part to buffer \( B_1 \). (State \( i \) must be such that the lead machine \( M_0 \) can produce a part.) We shall prove that:

\[ \text{if } \quad e(m) < \frac{1}{2} d(m) \quad (3.26) \]

\[ \text{then } \quad v_{i^*} > v_{i^*}^* \quad (3.27) \]

so that the optimal decision for state \( i \) is also to
route to \( B_1 \). (The opposite case, routing to buffer \( B_2 \),

can be handled similarly.) By simple algebra:

\[
v_i^* - v_i^* = v_i^*(m) - v_i^*(m) + (v_i^* - v_i^*(m)) + (v_i^*(m) - v_i^*)
\]

(3.28)

By (3.25) and (3.23):

\[
v_i^* - v_i^* = |v_i^*(m) - v_i^*(m)| + (v_i^* - v_i^*(m)) + (v_i^*(m) - v_i^*)
\]

(3.29)

\[
\geq d(m) + (v_i^* - v_i^*(m)) + (v_i^*(m) - v_i^*)
\]

(3.30)

Since \( a \geq -|a| \) for all real numbers, we have:

\[
v_i^* - v_i^* \geq d(m) - |v_i^* - v_i^*(m)| - |v_i^*(m) - v_i^*|
\]

(3.31)

By (3.24) and (3.26), then:

\[
v_i^* - v_i^* \geq d(m) - e(m) - e(m) > 0
\]

(3.32)
We conclude that $v_i^* > v_i^{*\prime}$, as claimed.

Unfortunately, it is very difficult to estimate the error $e(m)$ in the approximations $v_i(m)$. It is easy to measure how much the approximations change from one iteration to the next; for example, we can compute the maximum change per iteration $c(m)$:

$$c(m) = \max_i |v_i(m+1) - v_i(m)| \quad (3.33)$$

But this gives very little information about the actual error in $v_i(m)$. (In one case it was discovered that the actual error in $v_i(m)$ for one state $i$ was at least $3 \times 10^5$ times the maximum change per iteration.) Attempts to bound the error by monitoring the rate at which $c(m)$ decays to zero have likewise been unsuccessful. So, to ensure that truly optimal policies are found for the cases in this study, we continue the iterations until the maximum change per iteration $c(m)$ is very much smaller than the critical difference $d(m)$, by a factor of $10^{-10}$ to $10^{-15}$.

In a practical situation, we would never go to these lengths. By computing the Odoni bounds (Odoni [1969]), we can tell how far the production rate of the latest computed routing policy is from the optimal
production rate. When the production rate is sufficiently close to optimal, we would stop iterating. However, since a primary goal of this thesis is to describe the structure of exact optimal policies, we are forced to iterate beyond this practically reasonable limit. (Fortunately, the computer system used for this research offers a magnificent degree of precision — 59 decimal digits.)

There are algorithms for Markov decision problems with time-discounted rewards which give the error bounds on $v^*_i(m)$ necessary for a good stopping rule (Porteus and Totten [1978]). These algorithms involve reordering the states. This author is unaware of similar algorithms for problems without discounting.

3.7 Computational Complexity

The algorithm described in this chapter has been programmed in PL-1 on a Honeywell 6180 computer with the MULTICS (Multiplexed Information and Computing Service) operating system. Computations are done in decimal arithmetic, usually with a precision of 30 decimal digits.

In the author's experience with this program, using the very stringent convergence criterion described in
the previous section, the number of iterations required to determine the exact optimal routing policy is approximately equal to the number of states in the model. This is a very rough, order-of-magnitude estimate; the actual number of iterations varies greatly with the system parameters. The computational effort per iteration is also proportional to the size of the state space: with 30 decimal digits precision, the c.p.u. time is approximately 1 second per iteration per 2500 states. The total c.p.u. time, then, is roughly proportional to the square of the size of the state space or the fourth power of the buffer capacity (if \( N_1 = N_2 \)). This puts a hard limit on the size of problems which can be solved optimally using this program. For example, a problem with three unreliable machines and buffers of capacity 20 has about 3000 states and takes roughly one hour of c.p.u. time to solve. Extrapolating from our complexity measurements, we estimate that doubling the buffers would raise c.p.u. requirements to 16 hours. One should not conclude, however, that this algorithm is practically worthless or that the problem is insolvable. Things could be improved considerably by using binary instead of decimal arithmetic, by computing with less precision, and by stopping when the computed policy is nearly optimal in performance. (See section 3.6.)
Programming in FORTRAN on a faster computer system could also reduce c.p.u. time substantially.

It should also be mentioned that computer memory requirements are proportional to the size of the state space. With careful programming and a moderate degree of precision, the storage needs are not excessive.

3.8 Summary

D. J. White's method of successive approximations is presented in this chapter, along with a variety of improvements to speed convergence. The algorithm, as implemented for this thesis research, is described in detail, including rough estimates of computational complexity.
4. **A Heuristic Feedback Strategy**

In this chapter a heuristic feedback routing rule which is intuitively reasonable and extremely easy to implement is presented. This heuristic is useful for comparisons when the structure of the optimal strategy is examined in chapter 5. The heuristic and optimal rules are also compared with regard to performance in chapter 6, in order to determine when the heuristic works well. This is of practical significance, since it is much more difficult to determine and implement the optimal strategy.

In this chapter, we deal only with the continuous-time model of chapter 2. A completely analogous treatment of the discrete-time model can be performed, resulting in an identical heuristic strategy.

4.1 **The Proposed Heuristic: Worktime Balance**

The heuristic strategy routes parts in a way which tends to equalize the number of hours of work present in each network branch. Specifically, whenever a part is completed at the lead machine $M_0$, we compute the expected time until machine $M_1$ will finish its current workpiece and all the workpieces currently waiting in buffer $B_1$. This computation takes into account the
operational status $o_1$ of $M_1$, the buffer level $n_1$, and the processing, failure and repair rates ($\mu_1$, $p_1$ and $r_1$, respectively) of $M_1$. This quantity is compared with the corresponding expected worktime for machine $M_2$, and the part at $M_0$ is routed to the downstream branch with the smaller expected worktime.

This heuristic, called the worktime balance strategy, is based on the following intuitive reasoning. We would like to maximize the production rate of the manufacturing system. Production is lost due to starvation of downstream machines or blockage of the lead machine, so we must try to avoid these occurrences. To a great extent this is beyond our control, since there are many stochastic disturbances, and since we cannot influence the failure rates, repair rates, and processing rates. Routing decisions can make some difference, however. Focus first on the starvation problem. When both downstream machines have parts to process, the system is producing parts as fast as possible. When $M_1$ and $M_2$ are both starved, the system produces nothing, but intelligent routing in the immediate past could not have avoided this. However, if one downstream machine has workpieces in its buffer while the other is starving, then the instantaneous system production rate is not as
great as it might be (if the parts had been better distributed), and different routing decisions in the immediate past could have avoided this. Since the production which is lost under these circumstances cannot always be made up by future production, we would like to avoid such imbalances as much as possible. Therefore, whenever the lead machine $M_0$ completes a part, let us assume the worst: that it will be a very long time before $M_0$ completes its next part (due either to failures or an unusually long processing time), and that during this time one of the downstream machines will starve. We determine which machine this is likely to be and route the current part to that machine. Since it is very difficult to compute the probability that one downstream machine will starve before the other (see appendix 4 for the case of reliable machines), instead we compare the mean times till starvation. This is simply the worktime balance strategy described earlier.

Now consider the problem of blockage. Roughly speaking, blockage can be avoided by keeping the buffer levels as low as possible. This is done by keeping both downstream machines busy whenever possible, since the buffers are drained at the maximum rate when both machines are working. Thus the worktime balance heurs-
tic is seen to perform well with respect to both starvation and blockage.

It may seem that we have been arguing circularly. This is because the loss of production due to starvation and that due to blockage are closely linked, especially for a balanced system. As mentioned above, needless starvation of a downstream machine increases the likelihood of future blockage or the lead machine because the storage is not drained as fast as it might be. Moreover, this blockage of $M_0$ tends to produce future starvation of $M_1$ and $M_2$, since parts are not added to the storage as fast as they might be. By balancing the buffer levels, a good routing strategy will tend to keep both buffer levels closer to midrange, thereby reducing both starvation and blockage.

The heuristic just presented is very similar to one studied by Foschini for a slightly different system and a different objective (Foschini [1977]). Foschini's model (translated into our terms as explained in sections 1.2 and 2.1.4) has infinite buffers and a slow lead machine, so that blockage is impossible and production is never lost. The objective instead is to minimize the time an average workpiece spends in the system. Foschini's heuristic routes each part so as to minimize
its own expected delay. In other words, he uses a "customer-optimized" strategy as a heuristic for the "socially optimal" strategy of interest. Although the average delay per part is closely related to the production rate (Little [1961 a]), minimizing the average delay is not exactly the same as maximizing the production rate. Likewise, Foschini's heuristic for delay is similar but not identical to the worktime balance heuristic for production rate. Both strategies compare the expected number of hours of work in each downstream branch, but Foschini's formula includes the part just completed at $M_0$. It compares the expected time it would take $M_1$ to drain its buffer, if the part at $M_0$ were routed to $M_1$, with the expected time for $M_2$ to empty $B_2$, if the part at $M_0$ were routed to $M_2$. The production rate heuristic, because of a greater concern for starving machines than for delayed parts, makes its comparison based only on the parts already in the buffers. This difference in the two strategies is most evident if one downstream machine, say $M_1$, is several times faster and much more reliable than $M_2$, and when the buffers are empty. Under these circumstances, Foschini's delay heuristic will not route any parts to $M_2$ until there are several parts waiting in buffer $B_1$ for machine $M_1$. The production rate heuristic, on the other hand, will
always route a part to a starving machine $M_2$ unless $M_1$ is also starving.

4.2 **Mathematical Statement of the Heuristic**

In this section a precise statement of the worktime balance heuristic is given. The formulas are different for perfectly reliable downstream machines and unreliable downstream machines.

4.2.1 **Rule for Reliable Downstream Machines**

Define a function $T(n; \mu)$ to be the expected time for a perfectly reliable machine with processing rate $\mu$ to complete $n$ available workpieces, including one part currently being processed. Since the Poisson part completion process is memoryless, the expected time to complete the first part is simply $\frac{1}{\mu}$, regardless of the time already spent processing that part. Each subsequent part adds another interval of expected length $\frac{1}{\mu}$, so that:

$$T(n; \mu) = \frac{n}{\mu} \quad (4.1)$$

The worktime balance heuristic is expressed in terms of $T$ below and in the flowchart of figure 4.1:

- If the lead machine is under repair, it cannot complete a part, so no routing decision is needed (i.e., $k=0$).
- If both buffers are full, the lead machine is blocked. It cannot complete a part, so no routing decision is needed (i.e., k=0).

- If one buffer is full, route the next part to the other buffer.

- If neither buffer is full, use the following rule:
  
  Send the next part to buffer B₁ (i.e., k=1) if T(n₁; µ₁) < T(n₂; µ₂).
  
  Send the next part to buffer B₂ (i.e., k=2) if T(n₁; µ₁) > T(n₂; µ₂).
  
  Send the next part to the downstream machine with the greater processing rate µ in case of ties, i.e., if T(n₁; µ₁) = T(n₂; µ₂).

In short, the routing strategy attempts to drive the system to an equilibrium region of the state space described by the condition:

\[ T(n₁; µ₁) = T(n₂; µ₂) \]

(4.2)

or

\[ \frac{n₁}{µ₁} = \frac{n₂}{µ₂} \]

(4.3)

or

\[ \frac{n₁}{n₂} = \frac{µ₁}{µ₂} \]

(4.4)

Of course, this condition can rarely be met exactly, due
Figure 4.1. Worktime balance heuristic for perfectly reliable downstream machines.
to the state space boundaries, the discrete character of \( n_1 \) and \( n_2 \), and the various stochastic effects which make perfect control of the buffer levels impossible.

Notice which system parameters enter into the heuristic feedback law. The processing rates \( \mu_1 \) and \( \mu_2 \) of the downstream machines are very important, the critical relationship being their ratio, as shown in equation (4.4). The routing decisions do not depend in any way, however, on the parameters \( \mu_0 \), \( p_0 \) and \( r_0 \) of the lead machine. The buffer capacities \( N_1 \) and \( N_2 \) only have an effect at the upper state space boundaries: if \( n_1 = N_1 \) or \( n_2 = N_2 \), then the routing decision is forced. In the interior of the state space, however, the routing decisions are completely independent of \( N_1 \) and \( N_2 \).

Notice also how routing decisions depend on the state variables. The dependence on the lead machine state \( \sigma_0 \) is trivial, and the dependence on the buffer levels is through their ratio \( \frac{n_1}{n_2} \), as shown in equation (4.4), except at the upper state space boundaries.

It should be mentioned that Foschini's heuristic for minimizing delay compares \( T(n_1+1; \mu_1) \) with \( T(n_2+1; \mu_2) \) but is otherwise identical to the strategy outlined above. The corresponding equilibrium condition is:
\[
\frac{n_1 + 1}{\mu_1} = \frac{n_2 + 1}{\mu_2}
\]  
(4.5)

or

\[
\frac{n_1 + 1}{n_2 + 1} = \frac{\mu_1}{\mu_2}
\]  
(4.6)

4.2.2 *Rule for Unreliable Downstream Machines*

Define a function \(T(n, \sigma; \mu, p, r)\) to be the expected time for an unreliable machine to process \(n\) available workpieces, including one part which is already partially processed, given that the machine is currently in operational state \(\sigma\), where \(\mu\), \(p\) and \(r\) are respectively the processing, failure and repair rates of the machine. In appendix 2 we prove the following formula for \(T(1, 1; \mu, p, r)\), i.e., the expected time for an unreliable machine to complete the part it is currently processing, given that the machine is currently operational:

\[
T(1, 1; \mu, p, r) = \frac{r+p}{\mu r} = \frac{1}{\rho}
\]  
(4.7)

Recall that \(\rho\) is the isolated production rate of the machine. (In fact, in appendix 2 we derive not only the expected time, but also the entire conditional probability distribution.) Note that the time already spent
working on the part is irrelevant to the determination of the expected time till completion, due to our Markovian assumptions.

Since a machine is guaranteed to be operational at the instant it begins to process a new part, equation (4.7) also gives the expected total time that any part spends in the machine. This is not surprising, since this quantity is the reciprocal of the throughput \( \rho \) of the machine in isolation. We conclude that:

\[
T(n, 1; \mu, p, r) = \frac{n(r+p)}{\mu r} = \frac{n}{\rho} \quad (4.8)
\]

Now consider the case where the machine is currently under repair (\( \sigma = 0 \)). Since the expected repair time is \( \frac{1}{r} \), after which the machine is again operational, we have:

\[
T(n, 0; \mu, p, r) = \frac{1}{r} + T(n, 1; \mu, p, r) \quad (4.9)
\]

\[
= \frac{1}{r} + \frac{n(r+p)}{\mu r} \quad (4.10)
\]

\[
= \frac{1}{r} + \frac{n}{\rho} \quad (4.11)
\]

The formulas for \( T(n, 1; \mu, p, r) \) and \( T(n, 0; \mu, p, r) \) can be conveniently combined as follows:
The worktime balance heuristic for unreliable downstream machines is expressed in terms of this function $T$ below and in the flowchart of figure 4.2:

- If the lead machine is under repair, it cannot complete a part, so no routing decision is needed (i.e., $k=0$).

- If both buffers are full, the lead machine is blocked. It cannot complete a part, so no routing decision is needed (i.e., $k=0$).

- If one buffer is full, route the next part to the other buffer.

- If neither buffer is full, use the following rule:

  Send the next part to buffer $B_1$ (i.e., $k=1$) if $T(n_1, \sigma_1; \mu_1, p_1, \Gamma_1) < T(n_2, \sigma_2; \mu_2, p_2, \Gamma_2)$.

  Send the next part to buffer $B_2$ (i.e., $k=2$) if $T(n_1, \sigma_1; \mu_1, p_1, \Gamma_1) > T(n_2, \sigma_2; \mu_2, p_2, \Gamma_2)$.

  Send the next part to the downstream machine with the greater isolated production rate $\rho$ in case of ties, i.e., if $T(n_1, \sigma_1; \mu_1, p_1, \Gamma_1) = T(n_2, \sigma_2; \mu_2, p_2, \Gamma_2)$.

The heuristic routing strategy in the case of unreliable downstream machines attempts to drive the system to an equilibrium described by the following condition:
Figure 4.2. Worktime balance heuristic for unreliable downstream machines.
(4.14)\[ T(n_1, \alpha_1; \mu_1, p_1, r_1) = T(n_2, \alpha_2; \mu_2, p_2, r_2) \]
or

\[ \frac{(1-\alpha_1)}{r_1} + \frac{n_1(r_1+p_1)}{\mu_1 r_1} = \frac{(1-\alpha_2)}{r_2} + \frac{n_2(r_2+p_2)}{\mu_2 r_2} \] (4.15)

or

\[ \frac{(1-\alpha_1)}{r_1} + \frac{n_1}{\rho_1} = \frac{(1-\alpha_2)}{r_2} + \frac{n_2}{\rho_2} \] (4.16)

As we mentioned before, this condition can rarely be met exactly, due to the state space boundaries, the discrete character of \( n_1 \) and \( n_2 \), and the stochastic disturbances.

The dependence of the routing policy on the various system parameters and state variables is not as simple in the case of unreliable downstream machines as it is for perfectly reliable downstream machines, but a few remarks are in order. Notice that, as before, the parameters \( \mu_0 \), \( p_0 \) and \( r_0 \) of the lead machine have no effect on the routing rule and that, as before, the buffer capacities \( N_1 \) and \( N_2 \) influence routing decisions only at the upper state space boundaries. The dependence on \( \mu_1 \) and \( \mu_2 \) is not simply through their ratio, as is shown in equation (4.15).
Finally, we mention that the heuristic for minimizing delay compares $T(n_1+1, \sigma_1; \mu_1, p_1, r_1)$ with $T(n_2+1, \sigma_2; \mu_2, p_2, r_2)$ but is otherwise identical to the worktime balance heuristic. The corresponding equilibrium condition is:

$$\frac{(1-\sigma_1)}{r_1} + \frac{(n_1+1)(r_1+p_1)}{\mu_1 r_1} = \frac{(1-\sigma_2)}{r_2} + \frac{(n_2+1)(r_2+p_2)}{\mu_2 r_2} \quad (4.17)$$

or

$$\frac{(1-\sigma_1)}{r_1} + \frac{(n_1+1)}{\rho_1} = \frac{(1-\sigma_2)}{r_2} + \frac{(n_2+1)}{\rho_2} \quad (4.18)$$

4.3 Graphical Description of the Heuristic

When the worktime balance heuristic is graphed as explained in section 2.2.1, the resulting figure is very simple. Let us first look at the case of perfectly reliable downstream machines. An example with $N_1 = 5$, $N_2 = 10$, and $\frac{\mu_2}{\mu_1} = 3$ is shown in figure 4.3. The lead machine may be perfectly reliable or unreliable. In either case, we only need to show one state space rectangle, since routing decisions are not required ($k = 0$) for states where the lead machine is under repair ($\sigma_0 = 0$). Moreover, the routing decisions given by the
Figure 4.3. Graph of worktime balance heuristic for perfectly reliable downstream machines.
worktime balance heuristic for an unreliable but operational lead machine are exactly the same as those for a perfectly reliable lead machine, and these decisions do not depend on the parameters of the lead machine.

Figure 4.3 shows that, in the interior of the state space, the decision boundary is a straight line of slope \( \frac{\mu_2}{\mu_1} = 3 \). This line describes the equilibrium condition of equation (4.4), except that it is shifted slightly so that no point falls exactly on the boundary. This shift corresponds to the tie-breaking rule in our heuristic: that if \( T(n_1; \mu_1) = T(n_2; \mu_2) \), then the part should be routed to the faster machine.

The graph of Foschini's minimum delay heuristic for perfectly reliable downstream machines is similar to the worktime balance graph. Foschini's decision boundary has the same slope, but it is shifted vertically by an amount \( \frac{\mu_2}{\mu_1} - 1 \). (Compare equation (4.5) with (4.3).) This enlarges the decision region corresponding to the faster downstream machine. Figure 4.4 shows the delay heuristic for the same system parameters as figure 4.3.

Things are, of course, more complicated when the downstream machines are unreliable. Now there are four
Figure 4.4. Graph of minimum delay heuristic for perfectly reliable downstream machines.
state space rectangles of slightly different sizes, since transient states are not considered. Again, it does not matter whether the lead machine is reliable or unreliable, since decisions are only made when $\sigma_0 = 1$, and since the heuristic gives the same decisions for $M_0$ perfectly reliable as for $M_0$ unreliable but operational. Figure 4.5 is a graph of the worktime balance heuristic for the system parameters given in table 4.1. In the interior of each state space rectangle, the decision boundary is a straight line of slope $\frac{\rho_2}{\rho_1} = \frac{3}{2}$. This line describes the equilibrium condition of equation (4.15), but it is shifted slightly to break ties at points which exactly satisfy (4.16). (In case of ties, parts are routed to the machine with the greater isolated production rate $\rho$.) The graph of the delay heuristic is similar to the worktime balance graph, with decision boundaries of the same slope. The boundaries of the delay strategy are shifted vertically, however, by an amount $\frac{\rho_2}{\rho_1} - 1$. (Compare equation (4.18) with (4.16).) This enlarges the decision region corresponding to the more productive downstream machine.
\( N_1 = 6 \)
\( N_2 = 6 \)

\( \mu_0, p_0, \, r_0 \text{ unspecified} \)

\[
\begin{align*}
\mu_1 &= .5 \\
p_1 &= .1 \\
r_1 &= .2 \\
\mu_2 &= .8 \\
p_2 &= .15 \\
r_2 &= .25
\end{align*}
\]

\( \Rightarrow \rho_1 = \frac{1}{3} \)
\( \Rightarrow \rho_2 = \frac{1}{2} \)

Table 4.1. System parameters for figure 4.5.
Figure 4.5. Graph of worktime balance heuristic for unreliable downstream machines.
4.4 **Summary**

In this chapter the worktime balance routing strategy is presented as a heuristic for maximizing production rate. This strategy is justified on intuitive grounds and compared with a similar heuristic suggested by Foschini for minimizing workpiece delay. Explicit formulas for both heuristics are derived for the case of perfectly reliable downstream machines and also for unreliable downstream machines, and graphs of the strategies are examined.

These heuristics are useful for comparisons with the optimal routing policy in chapters 5 and 6. It is shown in chapter 6 that the production rates achieved by these heuristic control policies, particularly the worktime balance strategy, are nearly optimal, at least for those cases studied with perfectly reliable downstream machines. The heuristics are of practical importance because they are extremely easy to implement: the routing decision can be determined for each state individually by performing very simple algebraic operations on the state variable values. The optimal routing decisions, on the other hand, can only be found by solving the dynamic programming problem of chapter 2. This is
much more difficult, especially if the state space is large.
5. **STRUCTURE OF THE OPTIMAL ROUTING POLICY**

By means of the algorithm outlined in chapter 3, optimal routing policies were computed for the full system model and some simplified models, using many sets of system parameters. The results are described in the next two chapters. The performance of the optimal strategy is the subject of chapter 6. The structure of the optimal feedback law is examined in this chapter. We would like to show how the optimal routing decision depends on the various state variables and to determine which system parameters affect the control law and in what ways. In order to exhibit these features clearly, it is often necessary to push the parameter values to extremes. For example, we examine systems with very unreliable machines (33% downtime), extremely asymmetric systems ($\mu_1 = 30 \mu_2$), and highly unbalanced systems ($\rho_0 = 4(\rho_1+\rho_2)$ and $\rho_0 = \frac{1}{4}(\rho_1+\rho_2)$).

Those features of the optimal control which were found for all system models studied are described in section 5.1. In section 5.2 we focus on the effects of system symmetry and system balance, temporarily ignoring the machine reliability issue. The model used for this section has three perfectly reliable machines. We complicate the model (and enlarge the state space) in
section 5.3 by introducing failures at the lead machine. Finally, section 5.4 describes the few cases computed using the full system model with all machines unreliable. Not much was done with this model, primarily because the large state space is computationally burdensome, but also because the number of system parameters involved is so large that introducing them all and varying even a few of them enormously complicates the problem. Still, much more work could and should be done with this full model to discover the effects of downstream failures on the structure and performance of optimal routing rules.

The worktime balance heuristic and the minimum delay heuristic described in chapter 4 are useful as points of comparison for describing optimal strategies in this chapter.

The reader should bear in mind that all statements made about optimal routing policies in the next two chapters summarize observations of policies computed for many numerical examples. These statements are not the results of a theoretical analysis of the problem.
5.1 Properties of the Optimal Strategy for All Models

In this section we describe three properties of an optimal routing strategy which were found for all system models and all parameter values studied. These properties are also shared by the worktime balance and minimum delay heuristics.

First, the optimal decision regions in each state space rectangle are simply connected sets. This is certainly what one would expect. (The sets are generally not convex, however.)

Second, the optimal decision boundary between region 1 and region 2 in each state space rectangle is monotonically increasing when viewed either as a function of \( n_1 \) or of \( n_2 \). That is, the curve always moves upward and/or to the right. What this says about the optimal routing rule is that:

1) If the decision for state \((\bar{n}_1, \bar{n}_2; \bar{\alpha}_0, \bar{\alpha}_1, \bar{\alpha}_2)\) is 1, then so is the decision for all states \((\bar{n}_1, n_2; \bar{\alpha}_0, \bar{\alpha}_1, \bar{\alpha}_2)\) with \( \bar{n}_2 \leq n_2 \leq \bar{N}_2 \).

2) If the decision for state \((\bar{n}_1, \bar{n}_2; \bar{\alpha}_0, \bar{\alpha}_1, \bar{\alpha}_2)\) is 2, then so is the decision for all states \((n_1, \bar{n}_2; \bar{\alpha}_0, \bar{\alpha}_1, \bar{\alpha}_2)\) with \( \bar{n}_1 \leq n_1 \leq \bar{N}_1 \).
This feature is also intuitively reasonable.

The third property, while not counterintuitive, is rather surprising. The buffer capacities \( N_1 \) and \( N_2 \) have (almost) no effect on the optimal routing decisions at states where \( n_1 < N_1 \) and \( n_2 < N_2 \). That is, if we find the optimal strategies for two systems whose parameters are identical except for the buffer capacities \( N_1 \) and \( N_2 \), then the optimal decisions will be the same over that portion of the state space that both systems have in common, except at the upper state space boundaries, where the routing decisions are forced. This feature is illustrated in figure 5.1, where the graphs of hypothetical optimal strategies for three systems with different buffer sizes are superimposed. (Only one state space plane is shown.) Except at the upper and right-hand edges of the state space rectangles, the decision boundaries coincide.

Very rarely an exception to this rule is observed where the decision boundary intersects the state space boundary at \( n_1 = N_1 - 1 \) or \( n_2 = N_2 - 1 \). Sometimes the decision at a single state at this intersection point changes when the buffer capacity is increased. Figures 5.2 and 5.3 show this slight boundary effect for three systems with identical machine parameters but
Figure 5.1. Independence of optimal routing decisions and buffer capacities.
different buffer capacities. The parameters of the systems are given in table 5.1. In figure 5.2, graphs of the optimal policies for systems 1 and 2, over one state space rectangle, are superimposed. Figure 5.3 shows the optimal policies for systems 1 and 3 over another state space rectangle. The decisions at the states marked with a star do depend on the buffer capacity $N_x$. This boundary effect was observed only for systems with the machine parameters of table 5.1. For all systems studied with other machine parameters, the buffer capacities had no effect on the optimal decisions at interior states.

5.2 Simplest Model: All Machines Reliable

Many cases were studied using a simplified system model with three perfectly reliable machines. Not only does this keep the state space manageable, but it permits us to examine the effects of system symmetry and balance apart from the reliability aspects. Later, when we complicate the model, we can tell which features of the optimal solution are attributable to which features of the model.

Section 5.2.1 describes the general shape of the decision boundary when all machines are reliable. The
All machines unreliable

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_0$</td>
<td>20.0</td>
</tr>
<tr>
<td>$p_0$</td>
<td>2.5</td>
</tr>
<tr>
<td>$r_0$</td>
<td>5.0</td>
</tr>
<tr>
<td>$\mu_1$</td>
<td>31.0</td>
</tr>
<tr>
<td>$p_1$</td>
<td>2.5</td>
</tr>
<tr>
<td>$r_1$</td>
<td>5.0</td>
</tr>
<tr>
<td>$\mu_2$</td>
<td>9.0</td>
</tr>
<tr>
<td>$p_2$</td>
<td>2.5</td>
</tr>
<tr>
<td>$r_2$</td>
<td>5.0</td>
</tr>
<tr>
<td>System 1:</td>
<td></td>
</tr>
<tr>
<td>$N_1$</td>
<td>15</td>
</tr>
<tr>
<td>$N_2$</td>
<td>25</td>
</tr>
<tr>
<td>System 2:</td>
<td></td>
</tr>
<tr>
<td>$N_1$</td>
<td>10</td>
</tr>
<tr>
<td>$N_2$</td>
<td>20</td>
</tr>
<tr>
<td>System 3:</td>
<td></td>
</tr>
<tr>
<td>$N_1$</td>
<td>5</td>
</tr>
<tr>
<td>$N_2$</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5.1. System parameters for figures 5.2 and 5.3.
Figure 5.2. Exceptional case showing effect of buffer capacities on decisions.
MACHINE STATES
\[ a_0 = 1 \]
\[ a_1 = 0 \]
\[ a_2 = 0 \]

Figure 5.3. Exceptional case showing effect of buffer capacities on decisions.
intercept of the decision boundary (i.e., the place where it meets the state space boundary $n_1 = 0$ or $n_2 = 0$) is covered in section 5.2.2. Finally, we discuss the slope of the decision boundary for large buffer levels in section 5.2.3.

5.2.1 General Shape of the Decision Boundary

Figure 5.4 is a sketch of the optimal decision boundary for a perfectly reliable system, with noteworthy features exaggerated. Graphs of the worktime balance strategy and the delay heuristic are also shown for comparison.

The optimal boundary is in general a curve, not a straight line as are the boundaries for the heuristics. There are no inflection points in the curve; that is, it is not S-shaped.

The intercept of the optimal decision boundary is always on the axis corresponding to the faster downstream machine and lies between the intercepts of the two heuristics.

The slope of the decision boundary, measured at any point, is between 1 and (approximately) $\frac{\mu_2}{\mu_1}$, the slope of the decision boundaries for the heuristics. For low
Figure 5.4. Exaggerated graph of optimal strategy and heuristics for a perfectly reliable system.
buffer levels, the slope is closer to 1; then, as \( n_1 \) and \( n_2 \) increase, the slope shifts closer to \( \frac{\mu_2}{\mu_1} \). The curve appears to approach a linear asymptote for large buffer levels.

5.2.2 Intercept of the Decision Boundary

There are two ways to describe the intercept of a decision boundary for a system with reliable downstream machines. If we have an explicit formula for the boundary, such as equation (4.6), we can simply give the exact real value of \( n_1 \) (or \( n_2 \)) at which the decision boundary intersects the state space boundary \( n_2 = 0 \) (or \( n_1 = 0 \)). But if the policy is described only by specifying a decision at each state, then we must list the two integer buffer levels on either side of the intersection point. For example, the intercept of the worktime balance strategy for reliable downstream machines (figure 4.3) is always between 0 and 1 (denoted \([0,1]\)) and the intercept of figure 4.4 is between 2 and 3 (denoted \([2,3]\)). Physically, the intercept indicates the number of workpieces we would be willing to see waiting for processing by the faster downstream machine before we would choose to route a part to the slower, starving downstream machine.
The intercept of the optimal decision boundary is greater than \([0,1]\) only if the system is very asymmetric. The intercept is never greater than that of the minimum delay heuristic, which from equation (4.6) is seen to be:

\[
\frac{\max (\mu_1,\mu_2)}{\min (\mu_1,\mu_2)} - 1
\]  

(5.1)

Table 5.2 lists the optimal intercepts for different sets of system parameters, along with the intercepts given by the two heuristics.

One might suspect that in the low-buffer region of the state space, where starvation is imminent, the optimal policy would tend to be shortsighted — that the decisions would be responses to the immediate threat and would attempt to keep the production rate as high as possible in the near future. For comparison then, let us consider a third feedback heuristic, called the short-term strategy, whose routing decisions maximize the expected short-term production over the time it will take the lead machine to process another part. A formula for this routing rule is derived in appendix 3. Like the other heuristics, its decision boundary is a straight line, but with a different slope. The inter-
<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>INTERCEPTS</th>
<th>Optimal</th>
<th>Worktime Balance</th>
<th>Minimum Delay</th>
<th>Short-Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_0$</td>
<td>$\mu_1$</td>
<td>$\mu_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>23</td>
<td>17</td>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[0,1]</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
<td>10</td>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[2,3]</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>10</td>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[2,3]</td>
</tr>
<tr>
<td>80</td>
<td>30</td>
<td>10</td>
<td>[0,1]</td>
<td>[0,1]</td>
<td>[2,3]</td>
</tr>
<tr>
<td>40</td>
<td>35</td>
<td>5</td>
<td>[1,2]</td>
<td>[0,1]</td>
<td>[6,7]</td>
</tr>
<tr>
<td>40</td>
<td>31</td>
<td>4</td>
<td>[1,2]</td>
<td>[0,1]</td>
<td>[6,7]</td>
</tr>
<tr>
<td>40</td>
<td>36</td>
<td>4</td>
<td>[2,3]</td>
<td>[0,1]</td>
<td>[8,9]</td>
</tr>
<tr>
<td>40</td>
<td>31</td>
<td>2</td>
<td>[2,3]</td>
<td>[0,1]</td>
<td>[14,15]</td>
</tr>
<tr>
<td>40</td>
<td>31</td>
<td>1</td>
<td>[4,5]</td>
<td>[0,1]</td>
<td>[30,31]</td>
</tr>
</tbody>
</table>

Table 5.2. Intercepts for the various routing rules.
cept of this heuristic is given by:

\[
\frac{\ln \left( 1 + \frac{\mu_0}{\min(\mu_1, \mu_2)} \right)}{\ln \left( 1 + \frac{\mu_0}{\max(\mu_1, \mu_2)} \right)} - 1
\] (5.2)

This is a very good approximation to the optimal intercept and is included in table 5.2. Unfortunately, the short-term heuristic does not accurately predict the optimal decision at any point other than this intercept (not even at states where the buffer level corresponding to the slower machine is only 1). This routing strategy will not be mentioned further in this report; any references to closed-loop heuristics will mean the worktime balance and minimum delay policies.

5.2.3 Asymptotic Slope of the Decision Boundary

For high buffer levels, the optimal decision boundary asymptotically approaches a straight line whose slope is between 1 and (approximately) \( \frac{\mu_2}{\mu_1} \), the slope of the decision boundaries for the worktime balance and minimum delay heuristics. The asymptote is approached slowly if the system is very asymmetric, quickly if it is roughly symmetric. In fact, if \( \mu_1 \) and \( \mu_2 \) differ by less than a factor of 2, the entire decision boundary is
nearly linear.

If the system is approximately balanced (i.e., \( \mu_0 = \mu_1 + \mu_2 \)), then the asymptotic slope of the decision boundary is close to \( \frac{\mu_2}{\mu_1} \). (For a perfectly balanced system, the asymptotic slope may not be precisely between 1 and \( \frac{\mu_2}{\mu_1} \), but slightly to the other side of \( \frac{\mu_2}{\mu_1} \). For example, see figure 5.5 which is discussed below.) If the system is unbalanced in either direction \( (\mu_0 > \mu_1 + \mu_2 \text{ or } \mu_0 < \mu_1 + \mu_2) \), then the asymptotic slope is closer to 1; that is, the decision region corresponding to the slower downstream machine is enlarged. The greater the degree of imbalance, the closer the asymptotic slope is to 1, but even for extreme imbalances, the slope does not reach 1. This effect of system balance is clearly shown in figure 5.5, where the asymptotic slope is plotted as a function of \( \frac{\mu_0}{\mu_1 + \mu_2} \) for a system with \( \mu_1 = 3, \mu_2 = 1 \), and \( \mu_0 \) varying from 1 to 16. The symmetry of this graph is remarkable. An intuitive explanation of these features could give useful insights into this problem and other related problems; unfortunately, such an explanation eludes the author.
Figure 5.5. Asymptotic slope of optimal decision boundary vs. lead machine speed $\mu_0$. 

$$\frac{\mu_0}{\mu_1 + \mu_2} \quad \text{(LOGARITHMIC SCALE)}$$
It was necessary to use rather large buffers \(N_1 = 50, N_2 = 20\) for this part of the study in order to clearly see and measure the asymptotes of the decision boundaries. There is no guarantee that the decision boundaries would continue along the "asymptotes" if still larger problems were solved, but the slope of the boundaries appeared to stabilize before \(n_1 = 50\) and \(n_2 = 20\) for the cases computed.

5.3 Intermediate Model: Unreliable Lead Machine and Reliable Downstream Machines

In this section we introduce failures at the lead machine \(M_0\) to see how this affects the optimal routing policy. We are concerned with both aspects of the lead machine's reliability, as discussed in section 2.1.4 and appendix 2. The first aspect is the average rate of parts flow through \(M_0\), measured by its isolated production rate \(\rho_0\) (equation (2.2)), which is determined by its isolated efficiency \(e_0\) (equation (2.1)). But it is not sufficient to consider \(e_0\) as the single measure of reliability. The second measure is the variability of the parts flow through \(M_0\), which depends heavily on the repair rate \(r_0\). The effects of both aspects of reliability are considered below.
As was the case for the perfectly reliable system, the optimal decision boundary is a curve which appears to asymptotically approach a straight line as \( n_1 \) and \( n_2 \) increase. As before, the asymptotic slope is between \( \frac{\mu_2}{\mu_1} \), the slope of the worktime balance and minimum delay decision boundaries. In all the examples studied, \( \mu_1 \) and \( \mu_2 \) differed by no more than a factor of 3, so the intercept of the optimal decision boundary was \([0, 1] \) in all cases; this is to be expected, based on the findings of section 5.2.2.

If the lead machine is highly (but not perfectly) reliable, the optimal decision boundary has much the same shape as for the perfectly reliable case: the slope is closer to 1 for low buffer levels and moves closer to \( \frac{\mu_2}{\mu_1} \) for high buffer levels. However, if the repair and failure rates are small (reduced reliability in the second-moment sense) then the curve bows very slightly the opposite way; that is, its slope is closer to 1 for high buffer levels than for low buffer levels. Moreover, if the repair and failure rates are extremely small, the curve becomes S-shaped. This effect is illustrated in figure 5.6, where optimal policies are plotted for three systems whose parameters are given in
table 5.3. For each system, the lead machine has the same isolated efficiency, but \( r_0 \) and \( p_0 \) vary. The worktime balance heuristic, which is the same for all three systems, is also included in figure 5.6.

In section 5.2.3 we showed the effect of system balance on the asymptotic slope of the decision boundary: as system balance is disturbed, the asymptotic slope shifts from \( \frac{\mu_2}{\mu_1} \) toward 1. This same effect seems to be present if the lead machine is unreliable, but very few cases were tested. Now we can discuss the effect of lead machine reliability on the asymptote. If the lead machine is very unreliable in both senses (i.e., low isolated efficiency plus a highly variable parts flow due primarily to a low repair rate), then the asymptotic slope is close to \( \frac{\mu_2}{\mu_1} \) in spite of system imbalance. This finding is satisfying, since \( \frac{\mu_2}{\mu_1} \) is the slope of the worktime balance decision boundary, and the intuition behind this heuristic is strongly based on the presence of major stochastic disturbances at the lead machine. Figure 5.6 and table 5.3 also illustrate this feature, since \( \eta_0 \) has poor isolated efficiency (67\%) in all three systems and \( r_0 \) gets progressively worse. In systems with \( \epsilon_0 \) as high as 80\% or 90\%, this effect was
$M_0$ unreliable

$M_1$ and $M_2$ reliable

Case A  Case B  Case C
$P_0 = 30$  $P_0 = 2$  $P_0 = 0.1$
$r_0 = 60$  $r_0 = 4$  $r_0 = 0.2$

$\nu_0 = 90$
$\nu_1 = 10$
$\nu_2 = 30$

$N_1 = 20$
$N_2 = 55$

$\{ e_0 = 66.7\%$
$\rho_0 = 60 \}$

Table 5.3. Parameters for three systems in figure 5.6.
Figure 5.6. Worktime balance strategy (W) and optimal strategies for systems A, B, and C of table 5.3.
not observed.

5.4 **Complex Model: All Machines Unreliable**

Very few examples were computed using the full system model with three unreliable machines. In the cases studied, all three machines had the same failure and repair rates, with isolated efficiencies of 67%-83%. Machine processing rates were 3-16 times the failure rate and 2-8 times the repair rate. System asymmetries and imbalances were substantial in some cases, but not extreme (1.3 \( \leq \frac{\mu_1}{\mu_2} \leq 3.5 \) and \( \frac{1}{2} \leq \frac{\rho_0}{\rho_1 + \rho_2} \leq 1 \)). The buffer capacities were too small (\( \leq 20 \)) to clearly show the shapes of the optimal decision boundaries, which all appeared approximately linear.

One general property of the optimal routing strategy for the full system model was found. If we superimpose two state space planes which have \( \sigma_0 = 1 \) and which have the same value for one downstream machine state variable \( \sigma_1 \) and different values for the other downstream machine state variable \( \sigma_j \), then the optimal decision region \( j \) for the plane where \( \sigma_j = 0 \) will be wholly contained within the optimal decision region \( j \) for the plane where \( \sigma_j = 1 \). In terms of the optimal routing rule, this means that:
1) If the decision for state \((\hat{n}_1, \hat{n}_2; \hat{\alpha}_0, 0, \hat{\alpha}_2)\) is 1, then so is the decision for state \((\hat{n}_1, \hat{n}_2; \hat{\alpha}_0, 1, \hat{\alpha}_2)\).

2) If the decision for state \((\hat{n}_1, \hat{n}_2; \hat{\alpha}_0, \hat{\alpha}_1, 0)\) is 2, then so is the decision for state \((\hat{n}_1, \hat{n}_2; \hat{\alpha}_0, \hat{\alpha}_1, 1)\).

This finding is intuitively reasonable and is similar to the second property described in section 5.1.

The optimal decision boundaries were found to be very close to the worktime balance decision boundaries for balanced systems. When \(\rho_0\) was much smaller than \(\rho_1 + \rho_2\), the intercepts of the optimal decision boundaries nearly matched those of the worktime balance heuristic, but the optimal slope was shifted toward 1 (as was the case for the perfectly reliable system).

Many more cases need to be computed before anything conclusive can be said about the optimal policy for the full system model.

5.5 Summary

In this chapter, the structure of the optimal routing policy is described for several versions of our manufacturing system model. In all cases, the optimal
decision regions are simply connected sets, and the optimal decision boundaries are monotonically increasing when viewed either as functions of $n_1$ or $n_2$. Decision regions corresponding to a particular downstream machine are larger in those state space planes where that machine is operational than in those planes where the machine is under repair. The buffer capacities $N_1$ and $N_2$ have almost no effect on routing decisions in the interior of the state space (where $n_1 < N_1$ and $n_2 < N_2$).

The simplest system model, one with perfectly reliable machines, is studied in great detail. The optimal decision boundary in this case is a smooth curve with no inflection points. Its intercept is on the axis corresponding to the faster downstream machine. The intercept is larger for asymmetric systems and is approximated well by a routing rule based on short-term optimization. The slope of the optimal decision boundary is closer to 1 for low buffer levels and moves toward $\frac{\mu_2}{\mu_1}$ as $n_1$ and $n_2$ increase, eventually reaching a limit. This asymptotic slope is approximately $\frac{\mu_2}{\mu_1}$ only for systems which are nearly balanced.

When the model is enhanced to include failures at the lead machine, the shape of the decision boundary
changes. For extremely unreliable lead machines, inflection points are introduced, and the asymptotic slope is close to $\frac{\mu_2}{\mu_1}$ in spite of system imbalance.

The few cases computed for the complete system model, with three unreliable machines, showed that the worktime balance heuristic closely approximates the optimal strategy when the system is balanced.

In summary, it appears that the optimal routing policy looks very much like the worktime balance strategy if the system is nearly symmetric or if it is approximately balanced or if the lead machine is extremely unreliable. (This statement is merely a conjecture for systems with unreliable downstream machines.) Notice that this comparison concerns only the structure of the two policies, i.e., the shapes of their graphs. It says nothing about their performance, which is the subject of the next chapter.
6. PERFORMANCE OF VARIOUS ROUTING STRATEGIES

The performance of the optimal routing policy, two closed-loop heuristic rules, and an open-loop strategy are compared for a wide range of system parameters.

6.1 Outline of this Study

The performance of several routing policies is examined for a manufacturing system model with reliable downstream machines and an unreliable lead machine. The strategies compared are the optimal feedback law, the worktime balance heuristic, Foschini's minimum delay heuristic, and an open-loop, randomized policy which routes a part to downstream machine 1 with probability $\frac{\mu_1}{\mu_1 + \mu_2}$ when space is available in both buffers. (These are generally not the optimal routing probabilities for an open-loop, randomized rule; the optimal split should depend on all the system parameters, including the buffer capacities. The given rule seems reasonable, however, and is easy to program.)

The various routing strategies are compared for a wide range of system parameters. Nominal parameter values are listed in table 6.1. Notice that the nominal system is balanced, since $\rho_0 = \mu_1 + \mu_2 = 3.0$. The system
$M_0$ unreliable

$M_1$ and $M_2$ reliable

\[
\begin{align*}
\nu_0 &= 3.3 \\
p_0 &= 0.03 \\
r_0 &= 0.3 \\
\end{align*}
\Rightarrow \quad \begin{cases}
e_0 &= 90.9\% \\
\rho_0 &= 3.0
\end{cases}
\]

$\nu_1 = 1.0$

$\nu_2 = 2.0$

$N_1 = 5$

$N_2 = 10$

Table 6.1. Nominal system parameters for performance study.
was chosen to be rather asymmetric ($\mu_2 = 2\mu_1$) so that the three closed-loop routing rules would be substantially different. These three policies are graphed in figure 6.1.

A series of one-dimensional excursions are made from these nominal parameters. In section 6.2, the buffer capacities $N_1$ and $N_2$ are varied, subject to the constraint that $N_2 = 2N_1$. The system balance is altered in section 6.3 by changing the processing rate $\mu_0$ or the lead machine and leaving all other parameters fixed. The effect of lead machine reliability on system performance is examined in section 6.4 in three different ways. First the failure rate $p_0$ of the lead machine is varied, then the repair rate $r_0$ is varied. In each case the processing rate $\mu_0$ of the lead machine is adjusted to compensate for the variation in its isolated efficiency $e_0 = \frac{r_0}{r_0 + p_0}$, so that overall system balance is maintained. That is, the isolated production rate $\rho_0 = \mu_0 e_0$ of the lead machine is kept constant. Finally, we vary $p_0$ and $r_0$ together in fixed proportion, to see how the magnitude of these rates affects the system production rate. Recall from section 2.1.4 that the isolated efficiency $e_0$ and the isolated production rate $\rho_0$, the first-moment measures of machine reliability,
Figure 6.1. Optimal policy and two closed-loop heuristics for system described in table 6.1.
depend only on the ratio of $p_0$ to $r_0$. By varying $p_0$ and $r_0$ together we can change the variability of the flow through an (isolated) lead machine without changing its mean rate. (See appendix 2.)

Several other meaningful variations could be performed: for example, changing $p_0$ or $r_0$ without maintaining system balance, or altering the ratio of the downstream processing rates $\mu_1$ and $\mu_2$. The experiments conducted in the next few sections, however, are thought to be the most basic and the most interesting.

The emphasis of this study is on the relative performance of the different routing policies rather than on their absolute performance. We would like to determine the conditions under which feedback control is a significant improvement over open-loop control and also to find out when the optimal closed-loop policy is substantially better than the closed-loop heuristics.

6.2 Effect of Buffer Capacity

The performance of the four routing strategies is examined as the buffer capacity $N_1$ varies from 2 to 15 and $N_2$ varies from 4 to 30. (The 1:2 ratio of $N_1$ to $N_2$ is maintained.) All other system parameters retain the nominal values of table 6.1.
The (absolute) production rate of the manufacturing system under the four routing laws is graphed in figure 6.2. As one would expect, the performance under any policy improves as the buffers grow, and this improvement is greatest when the buffers are small. The curves for the optimal policy and the worktime balance heuristic are indistinguishable; these strategies have identical performance to four significant digits. The delay heuristic is very slightly worse, and the open-loop policy is noticeably poorer.

In order to better compare the routing strategies, the production rates of the three suboptimal rules can be divided by the optimal production rate. The resulting relative production rates are plotted in figure 6.3. We see that the (percentage) difference in the performance of various policies is greater when the buffers are small. But be careful to notice the vertical scale: even the open-loop policy is only about one percent below optimal.

6.3 Effect of System Balance

In the next set of trials, the balance of the system is disturbed by varying the processing rate \( \mu_0 \) of the lead machine from 2.0 to 6.0. The other parameters
Figure 6.2. Effect of buffer capacity on absolute performance.
Figure 6.3. Effect of buffer capacity on relative performance.
are fixed at the nominal values of table 6.1.

The absolute production rates of the four policies are graphed in figure 6.4. Naturally, system performance is better for larger $\mu_0$. Recall that the system is balanced when $\mu_0$ is 3.3 (since then $\rho_0 = 3.0 = \mu_1 + \mu_2$). If $\mu_0$ is less than about 2.6, then the lead machine is the bottleneck in the system, and the production rate under any policy is nearly equal to the isolated production rate $\rho_0$ of the lead machine. (For comparison, $\rho_0$ is shown as a dashed curve in figure 6.4.) As $\mu_0$ increases, the rate of improvement decreases, and performance levels off for $\mu_0$ greater than about 6.0, because the downstream machines together become the bottleneck at that point. (For comparison, $\mu_1 + \mu_2 = 3.0$ is shown as a dashed line in figure 6.4.)

The performance ranking of the four strategies is the same as in the preceding section. The worktime balance heuristic is as good as the optimal policy, to four significant digits. The delay heuristic is very slightly worse; still, its performance is optimal to three significant digits. Only the open-loop rule is significantly poorer: in some cases it is more than one percent below optimal.
Notice from figure 6.4 that the various strategies differ most in performance when the system is approximately balanced. With a major imbalance in either direction ($\mu_0$ too small or too large), even the open-loop control works extremely well. This feature is even more evident from figure 6.5, where the relative production rates are plotted. The reader is cautioned again to check the scales: the vertical deflections are actually quite small, and the range over which $\mu_0$ is varied is rather wide. The intent is to detect and exaggerate the trends by pushing the system parameters to extremes.

6.4 **Effect of Lead Machine Reliability**

The reliability of the lead machine is varied in three different ways to study its effect on the performance of the routing strategies. The isolated efficiency $e_0 = \frac{r_0}{r_0 + p_0}$ of the lead machine can be changed by varying the failure rate $p_0$ or the repair rate $r_0$; both of these effects are tested. In order to maintain the balance of the system, the processing rate $\mu_0$ of the lead machine is adjusted to offset the changes in its isolated efficiency $e_0$. That is, the isolated production rate $\rho_0 = \mu_0 e_0$ of the lead machine is maintained at its nominal value of 3.0. The other system parameters
Figure 6.4. Effect of system balance on absolute performance.
Figure 6.5. Effect of system balance on relative performance.
also retain their nominal values (table 6.1).

The effect of varying the lead machine efficiency is much the same whether this is done by changing $p_0$ or $r_0$. Figures 6.6 and 6.8 respectively show the absolute production rates for these experiments. As before, the performance curves for the worktime balance heuristic match the optimal curves to four digits, with the delay heuristic not far behind (three-digit agreement), and the open-loop performance lags by about one percent. From figures 6.6 and 6.8 we observe a marked improvement with increasing efficiency, in spite of the fact that the processing rate $\mu_0$ is being reduced as the failure rate $p_0$ decreases or the repair rate $r_0$ increases. The improvement is approximately linear, with a production rate increase of 0.3-0.7 percent for each one percent increase in $e_0$ (and corresponding one percent decrease in $\mu_0$).

The relative production rates, however, do not depend much on the lead machine's reliability, as can be seen from figures 6.7 and 6.9. The relative performance of the open-loop policy is somewhat better when the lead machine is very unreliable, but the other curves are flat.
Figure 6.6. Effect of lead machine reliability (failure rate) on absolute performance.
<table>
<thead>
<tr>
<th>Failure Rate $p_0$</th>
<th>.10</th>
<th>.06</th>
<th>.03</th>
<th>.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Rate $\mu_0$</td>
<td>4.0</td>
<td>3.6</td>
<td>3.3</td>
<td>3.1</td>
</tr>
<tr>
<td>Efficiency $e_0$</td>
<td>75%</td>
<td>83.3%</td>
<td>90.9%</td>
<td>96.8%</td>
</tr>
</tbody>
</table>

**Figure 6.7.** Effect of lead machine reliability (failure rate) on relative performance.
Figure 6.8. Effect of lead machine reliability (repair rate) on absolute performance.
Figure 6.9. Effect of lead machine reliability (repair rate) on relative performance.
In the third test of reliability, the isolated efficiency of the lead machine is held constant by fixing the ratio of the failure and repair rates, but the magnitudes of both rates are varied proportionally. As explained in section 2.1.4, a machine with very low failure and repair rates is very unreliable in a second-moment sense. Though its average production rate in isolation may be good, the parts flow is so variable that, when the machine is placed in our manufacturing system, the finite buffers cannot effectively smooth its output.

The absolute production rates are plotted in figure 6.10. As expected, a machine with high failure and repair rates performs better in the network than a machine which fails rarely but takes a long time to repair. However, the relative production rates, as before, are insensitive to this aspect of reliability. Their graphs, shown in figure 6.11, are very flat.

6.5 Summary

In this chapter the performance of the optimal routing policy is compared with three suboptimal policies over a wide range of system parameters. A model with perfectly reliable downstream machines and an
Figure 6.10. Effect of lead machine reliability (failure and repair rates) on absolute performance.
Figure 6.11. Effect of lead machine reliability (failure and repair rates) on relative performance.
unreliable lead machine is used. Parameters are given
nominal values, and several one-dimensional parameter
variation studies are conducted. For the nominal system
studied, we can make the following statements:

1) The performance of the worktime balance
heuristic is optimal to four significant
digits, and the minimum delay heuristic per-
forms optimally to three digits.

2) The production rate under a reasonable (but
not optimal) open-loop strategy is about one
percent below that attainable by optimal feed-
back control. (The greatest difference found
was 1.6%.)

3) The (percentage) difference in the performance
of the various strategies is greatest if the
buffers are small, the system is balanced, and
the lead machine is very reliable.

Numerical experiments should be conducted for other nom-
inal parameter sets to determine whether these state-
ments hold in general. For example, it is possible that
for some parameter values the minimum delay heuristic is
better than the worktime balance strategy. More impor-
tantly, a performance comparison study similar to this
one should be done for a model with unreliable downstream machines. It is quite possible that differences in the performance of the routing policies are much more pronounced if the downstream machines are unreliable. This is especially likely when comparing open-loop routing with feedback control, since downstream failures increase the uncertainty in the state of the system we are trying to control.
7. SUGGESTIONS FOR FUTURE RESEARCH

This chapter outlines areas for future research. Section 7.1 summarizes remaining work needed to complete the study begun in this thesis. Variations and extensions of this dynamic routing problem are discussed in section 7.2. Simple stochastic scheduling problems which could be attacked using similar models and methods are presented in section 7.3. The reader is reminded of the major limitation of this whole approach -- the size of the state space, Bellman's "curse of dimensionality" -- in section 7.4.

7.1 Remaining Work on this Problem

The most important area left unfinished in this thesis is the study of the structure and performance of the optimal routing policy for the full system model with three unreliable machines. Because there are three machine state variables (and thus eight state space planes), it is impractical to compute optimal strategies for systems where \(N_1 \cdot N_2\) is much larger than 250; the state space becomes unmanageably large. But with such small state space rectangles, it is difficult to detect curves or measure slopes. Consequently, the optimal decision boundaries cannot be studied at the level of
detail possible for the simpler models. Still, some work can be done toward describing the structure of the optimal strategy for the full model. Moreover, the size of the state space is less of an impediment to a study of the performance of the optimal control law, because the iterative algorithm of chapter 3 converges to the optimal production rate (to a reasonable degree of precision) long before the exact optimal policy is guaranteed to be found. As explained in chapter 6, we conjecture that the difference in performance of the closed-loop strategies (optimal or otherwise) and the open-loop, randomized strategy will be considerably greater when failures of the downstream machines are included in the model.

Work is also needed in developing some intuitive understanding of the properties discovered and described in chapters 5 and 6. Most baffling is the effect of the lead machine speed on the asymptotic slope of the optimal decision boundary for the perfectly reliable system (section 5.2.3).

Improvements to the algorithm would also be welcome. Perhaps White's algorithm could be modified to incorporate the lopsided simultaneous iteration method of Jennings and Stewart [1975]. (See also Stewart
There are other methods of solving Howard's equations (2.26) which could be tried: for example, the policy iteration algorithm of Howard [1960], linear programming (Shapiro [1979]), and the dual algorithm of Varaiya [1978]. (See also Popyack, Brown and White [1979].) Howard's algorithm would only be feasible if it were adapted to take explicit advantage of the block tridiagonal structure of the transition rate matrix, described in appendix 1. Another possibility is to apply a very small discount factor to future rewards in the model. Discounted problems are sometimes easier to solve. For example, Porteus and Totten [1978] describe iterative algorithms with guaranteed stopping rules, but these only apply to discounted problems.

Approaches to the approximate solution of very large problems should be suggested, examined and compared. The worktime balance heuristic is always a simple solution. As a second example, one could optimally solve an aggregated model where several workpieces are lumped together and modeled as one. The machine processing rates and buffer capacities would be scaled down accordingly, but the machine failure and repair rates would be unchanged. Schick and Gershwin [1978] find
that this technique, which they call a "delta transformation," works very well for approximating the production rate of a large-buffer manufacturing model similar to ours, but without routing choices. (For additional analysis, see Gershwin and Schick [1980].)

A third possibility is to model the level or material in the buffers as a continuous quantity and consider flows of material through the machines. Tsitsiklis [1980] is currently studying this model. Unfortunately, it appears that this problem can only be solved numerically, by discretizing the state space. However, the discretization need not be uniform. Tsitsiklis suggests using a fine mesh near the decision boundary and a coarse grid away from the boundary.

A fourth approach depends on a property of the optimal solution described in section 5.1: that decisions in the interior of the state space depend very little on the buffer capacities. Therefore, to approximate the optimal solution of a problem with large buffers, we could optimally solve the same problem with smaller buffers, use these computed decisions in those corners of the larger state space where the buffer levels are low, and use any reasonable closed-loop strategy over the rest of the state space. For example, we could
simply extend the computed decision boundaries with straight lines of slope \( \frac{\rho_2}{\rho_1} \), as suggested by the worktime balance and minimum delay heuristics.

A fifth possibility for large problems is to adapt the algorithm currently being developed by Santana and Platzman to find the optimal "regular" policy, that is, the best policy with a straight-line decision boundary (Santana and Platzman [1979], Santana [1980]).

These various approaches should be compared not only in terms of performance, but also with regard to ease of computation and implementation. The decisions given by any straight-line policy, such as the worktime balance strategy, the minimum delay heuristic, or the Santana-Platzman rule, could probably be computed in real time with a minimal amount of stored data. The other suggested policies must be precomputed and stored, either by listing the decisions at all states or by listing the states along the decision boundaries. This may or may not be convenient to implement, depending on the size of the state space and the computational environment of the actual manufacturing system.

Future research efforts could also be directed to the study of other closed-loop and open-loop heuristics.
For instance, a routing rule which compares the probabilities of each downstream machine starving first (rather than their mean times till starvation) is difficult to work with but would be interesting to study. Formulas for this rule for the case of perfectly reliable downstream machines are derived in appendix 4. Another possibility is to determine the optimal open-loop, randomized strategy. (The open-loop strategy used for the performance comparisons of chapter 6 did not use the optimal routing probabilities.) Or one could study a limited feedback routing rule, where the routing controller knows not only when a buffer is completely full, but also when one is completely empty, and makes the routing decision accordingly; when both buffers are in mid-range, a randomized rule is used.

As a final suggestion, it would be interesting to compare the optimal production rate of our two-buffer system with the production rate of a similar system shown in figure 7.1. Here the two buffers are merged into a single buffer from which both downstream machines draw parts. For any given amount of total storage $N_1 + N_2$, the one-buffer system should perform better than the two-buffer system. It would be useful to quantify this advantage.
Figure 7.1. A manufacturing network with a single, shared buffer.
7.2 Variations and Extensions of this Problem

Two variations of the problem studied in this thesis are mentioned in section 7.1: finding the optimal open-loop randomized strategy, and using a continuous flow model. Another possibility is to model the processing times as Erlang-distributed random variables, following Gershwin and Berman [1978] and Berman [1979], though this requires an additional state variable at each machine. One might also consider using a different control objective, such as the average delay per part (though it is not clear that this performance measure can be optimized in a standard Markovian decision framework), or optimizing some combined measure of throughput and average delay, or maximizing throughput subject to a constraint on average delay. Another approach is to include a penalty for in-process inventory and allow the lead machine to be idle even when space is available in the buffers.

Still another variation is to adapt the model to more closely approximate a synchronous transfer line. (See Schick and Gershwin [1978] for a synchronous model which could be modified to include parallel machines and routing decisions.) In such a system, the downstream machines are identical and have a deterministic, fixed
processing time per part. The lead machine also has a fixed cycle time which is exactly half that of the downstream machines, and all machines operate synchronously. A discrete-time model is appropriate for this system, where the time step is equal to the cycle time of either the lead machine or the downstream machines. The former approach is more precise; for example, it permits the routing of each part immediately upon completion of processing at the lead machine. But this model requires an extra binary state variable at each downstream machine to indicate whether the machine is currently performing the first or second half of the operation required for the current part. The latter approach is therefore recommended. During one time period of this latter model, each operating downstream machine produces one part, and the lead machine (if it is operating) produces two parts, which are routed at the same time. Both parts may be routed to the same downstream machine or they may be split between the downstream machines.

Finally, more complex networks could be tackled using extensions of the manufacturing model in this thesis and using the same algorithm. For example, routing policies for the multiple branch network of figure 7.2 could be studied. We could also imbed our two-stage
network in a variety of three-stage networks as shown in figure 7.3. The networks shown are the simplest to study, because each has only three buffers and a single point where routing decisions need to be made.

7.3 Stochastic Dynamic Scheduling Problems

There are several stochastic dynamic scheduling problems which are very similar to the dynamic routing problem studied in this thesis. Consider a manufacturing system with the layout depicted in figure 1.1, but which produces two types of parts. Type 1 parts must be processed first by the lead machine \( M_0 \) and then by downstream machine \( M_1 \). Type 2 parts are processed by \( M_0 \) then \( M_2 \). When \( M_0 \) completes a part, it must decide which type of part to process next. Such a system can be modeled as in chapter 2, but an extra binary state variable is needed at the lead machine to indicate the type of part currently being processed. This extra state variable also gives us the option of letting the processing, failure, and repair rates of the lead machine depend on the part type. As with the single-part problem, a policy which maximizes the steady-state throughput could be computed. Alternatively, we could seek to maximize a weighted sum of the production rates of the two part types. More complex objectives are
Figure 7.2. A multiple branch network.
Figure 7.3. Several simple three-stage systems with routing choices.
possible, such as maximizing the production rate subject to some penalty or constraint on the ratio of the different types of parts produced. Such a constraint arises in systems where parts of different types are processed separately and later assembled together. This would require extra state variables, however, to keep track of the number of parts of each type the system has already produced, or at least to count the mismatched parts.

The simplest such problem is to maximize the production rate subject to a 1:1 parts-ratio constraint. To satisfy this constraint, we insist that, at any time, the total number of type 1 parts produced by the system up to that time may not differ by more than a fixed number $N_3$ from the total number of type 2 parts produced up to that time. This problem requires a single extra state variable $n_3$ to denote the current production imbalance. We let $n_3$ take on values from $-N_3$ to $+N_3$, positive values indicating an excess of type 1 parts and negative values indicating an excess of type 2 parts. It is as though we had the imaginary three-buffer system shown in figure 7.4, where buffer $B_3$ of capacity $N_3$ can hold parts of either type. When a type 1 part meets a type 2 part in $B_3$, they are instantaneously assembled
Figure 7.4. Modeling a parts-ratio constraint using one extra state variable.
and removed from $B_3$, so $B_3$ will always contain parts of a single type. The state variable $n_3$ indicates the level of buffer $B_3$, positive if $B_3$ contains type 1 parts, negative if $B_3$ contains type 2 parts. When $B_3$ is full of type 1 parts, machine $M_1$ is blocked; when $B_3$ is full of type 2 parts, machine $M_2$ is blocked. Please note that $B_3$ is not meant to model a real storage element, but to illustrate an operating constraint we choose to impose on our manufacturing system.

More generally, let us assume that it is desired to produce type 1 parts and type 2 parts in the ratio $\gamma:d$, where $\gamma$ and $d$ are relatively prime integers. The most straightforward way to model this problem is to introduce two extra state variables $n_4$ and $n_5$ representing the number of unmatched parts of types 1 and 2 respectively. These quantities could be individually constrained:

$$0 \leq n_4 \leq N_4 ; \quad 0 \leq n_5 \leq N_5 \quad (7.1)$$

That is, we could have two output buffers $B_4$ and $B_5$ to hold unmatched parts of the two types, as shown in figure 7.5. When $\gamma$ parts of type 1 and $d$ parts of type 2 are available, they are instantaneously removed from the
Figure 7.5. Modeling a parts-ratio constraint using two extra state variables.
buffers and assembled. This introduces an awkward state constraint:

\[(n_4 \geq \gamma) \implies (n_5 < d)\] (7.2)

The feasible combinations of \(n_4\) and \(n_5\) lie in an L-shaped region of the \((n_4, n_5)\) plane, as shown in figure 7.6 for \(\gamma = 3\), \(d = 5\), \(N_4 = 8\), and \(N_5 = 10\).

Alternatively, as in the case of a 1:1 parts-ratio constraint, it is possible to compactly represent the current production imbalance by a single extra state variable \(n_3\) defined as follows:

\[n_3 = dn_4 - \gamma n_5\] (7.3)

We constrain \(n_3\) to be between \(-N_3\) and \(+N_3\). Again, we can consider this to be the level of the imaginary buffer \(B_3\) of figure 7.4. Whenever \(M_1\) completes a part, this buffer level rises by \(d\); when \(M_2\) completes a part, the level drops by \(\gamma\). (A basic result in number theory guarantees a one-to-one correspondence between pairs \((n_4, n_5)\) of non-negative integers satisfying constraint (7.2) and integers \(n_3\) satisfying the linear Diophantine equation (7.3). See Long [1972], p.95.) Figure 7.7
Figure 7.6. Feasible combinations of $n_4$ and $n_5$ for the model of figure 7.5.
Figure 7.7. Production imbalance $n_3$ corresponding to pairs $(n_4, n_5)$. 
shows the values of $n_3$ corresponding to the points $(n_4, n_5)$ of figure 7.6.

The basic scheduling model discussed throughout this section (i.e., the system of figure 1.1 with two part types) is of interest for another reason as well. This model is equivalent in the sense of Ammar [1980] and Ammar and Gershwin [1980] to the dual model described in section 1.2 and diagrammed in figure 1.2. In this dual model, machines $M_1$ and $M_2$ process parts, placing them in buffers $B_1$ and $B_2$ respectively, while machine $M_0$ can choose a part from either buffer. An optimal scheduling policy for the original system of figure 1.1 (with two part types) also gives, when suitably transformed, an optimal scheduling policy for the corresponding dual system of figure 1.2. This equivalence has not been proven, but it is a very strong conjecture. (See Ammar [1980] and Ammar and Gershwin [1980] for equivalence proofs for queueing models similar to ours, but without routing choices.)

Figure 7.8 shows the network layout for another stochastic scheduling problem which could be studied with models and methods similar to those of this thesis. Two part types are produced, each of which requires processing at both machines $M_1$ and $M_2$. While waiting for
Figure 7.8. A stochastic scheduling problem with two decision points.
processing by $N_2$, the parts are stored in buffers of finite capacity. The capacity constraints may be individual ($n_1 \leq N_1$, $n_2 \leq N_2$) or joint ($n_1 + n_2 \leq N$). Each machine must decide which type of part to process next, the objective being to maximize some measure of the system production rate. The processing, failure and repair rates of the machines could depend on the part type. This manufacturing system can be modeled with six state variables: two binary variables giving the operational status of the machines, two binary variables indicating the part type at each machine, and the two buffer levels. Additional state variables are needed if the ratio of parts produced of different types is important. Gershwin and Ammar [1979] have done some preliminary work on this system under open-loop, randomized control. The feedback control problem has not yet been addressed.

7.4 Considerations of State Space Size

Many of the problems suggested in sections 7.2 and 7.3 will be difficult to solve in the same manner as the problem addressed in this thesis, because the large number of state variables required for interesting problems results in an enormous state space. Even the simple problem studied in this thesis was close to the limit of practical computability when the machines were
unreliable and the buffer capacities were 20-50. As another example, a three-way branch problem (figure 7.2) with all machines unreliable and moderate buffer capacities (say 10) would have approximately 16,000 states and would take an enormous amount of computation time to solve optimally.

There are several ways to deal with this difficulty. One way is to trade off the different state variables: problems with large buffer capacities can be solved if most machines are perfectly reliable; systems with many unreliable machines can be handled if the buffers are small. Another approach is to surrender: instead of computing truly optimal strategies, we may have to be content with constructing and testing heuristics. It is relatively easy to compute the performance of any given control policy. It is also relatively easy, using D. J. White's algorithm and the Odoni bounds, to accurately estimate the optimal performance and to compute nearly-optimal policies which could be compared with the heuristics. (See section 3.6.) Future work on larger or more complex systems must either focus on such performance comparisons of heuristics or turn toward the development of more aggregated models.
8. SUMMARY AND CONCLUSIONS

This thesis is a study of dynamic routing strategies in a manufacturing network with a simple branching layout, unreliable heterogeneous machines (servers), and finite storage buffers (queues). The system is modeled as a Markov decision process, and the problem of finding a stationary feedback routing law to maximize the steady-state production rate is formulated as a stochastic dynamic programming problem. A variation of D. J. White's method or successive approximations is used to numerically compute optimal controls for wide ranges of the system parameters. The structure and performance of the optimal feedback policy are studied and compared with open-loop and closed-loop heuristics.

The more significant heuristics studied are the worktime balance strategy, which routes a part to the downstream machine with the minimum expected number of hours of waiting work, and the minimum delay strategy suggested by Foschini [1977], which routes each part so as to minimize the expected delay that part will experience in the system. These routing rules are very similar, and explicit formulas are derived for them in terms of the mathematical model of the manufacturing system. These strategies are of practical importance because
their routing decisions are extremely simple to compute, and because it is easy to extend these strategies to multiple-branch networks.

It is found, at least for the case of perfectly reliable downstream machines, that the optimal policy is close to the worktime balance strategy if the downstream machines are similar, or if the system is approximately balanced, or if the lead machine is extremely unreliable. As predicted by the worktime balance heuristic, the optimal decisions for system states in the interior or the state space are nearly independent of the buffer capacities. The worktime balance rule roughly predicts the effect of system asymmetries on the optimal strategy (so long as the system is balanced): both control policies try to maintain an equilibrium condition in which the ratio of the buffer levels is approximately equal to the ratio of the speeds of the downstream machines. Under the heuristic control law, the speed of the lead machine has no effect on the routing decisions. This is not the case, however, for the optimal control law. If the lead machine is reasonably reliable but is either very slow or very fast, so that the system is unbalanced, then the optimal routing strategy is more favorable to the slow downstream machine than is the worktime
balance strategy.

As far as performance is concerned, the worktime balance heuristic is excellent, at least for the cases studied with perfectly reliable downstream machines. In every example computed, the performance of this heuristic matches the optimal performance to four significant digits. The minimum delay heuristic is only slightly worse. Even an open-loop control law works well unless the buffers are small, the system is roughly balanced, and the lead machine is very reliable. (In the worst computed case, the performance of an open-loop heuristic was 1.6% below optimal.) However, to properly quantify the importance of feedback for this problem, it will be necessary to repeat this performance comparison study using unreliable downstream machines.

The approach taken in this thesis -- numerically computing exact optimal policies for detailed, discrete dynamical system models -- is not practical for problems much larger or more complex than the one addressed here. A few problems of similar size which could possibly be handled this way have been identified: an adaptation of this model to a synchronous transfer line and some stochastic dynamic scheduling problems with two part types. Future work on more complex systems should concentrate
on performance comparisons of heuristic strategies and/or the development of more aggregated dynamical models. Detailed models of small systems, such as the one studied in this thesis, might prove useful for validating these newer, streamlined models.
APPENDIX 1

STRUCTURE OF THE TRANSITION RATE MATRIX

This appendix describes the structure of the transition rate matrix for our manufacturing system model. The approach here closely follows the analysis by Schick and Gershwin [1978, pp. 83-111] of an unreliable transfer line model with no routing flexibility. Specifically, the transition rate matrix for our model is very similar in structure to the transition probability matrix for the 3-machine, 2-buffer version of the Schick and Gershwin model.

Let us order the states \((n_1, n_2, \sigma_0, \sigma_1, \sigma_2)\) of our manufacturing system lexicographically (i.e., from \((0, 0, 0, 0, 0)\) to \((N_1, N_2, 1, 1, 1)\), with the rightmost state variable \(\sigma_2\) varying fastest, and the leftmost state variable \(n_1\) varying slowest). To simplify matters, include the transient states in the list. With this ordering, the transition rate matrix \(A^K\) defined in section 2.2.2 exhibits the block-tridiagonal structure shown in figure A1.1. The matrix \(A^K\) is composed of \((N_1+1)^2\) submatrices, each of which has dimensions \(8(N_2+1) \times 8(N_2+1)\). The \((i,j)\)th block of \(A^K\) contains the rates for transitions from states where \(n_1 = i\) to states
Figure A1.1. Structure of the Transition Rate Matrix $A^K$. 

$A^K = \begin{array}{cccc}
Y^K_0 & Z^K_0 \\
\times & Y^K_1 & Z^K_1 \\
\times & Y^K_2 & Z^K_2 \\
\times & Y^K_3 & Z^K_3 \\
\times & Y^K_4 & \ddots \\
\vdots & \vdots & \ddots & Z^K_{N_1-1} \\
\times & Y^K_{N_1} \\
\end{array}$
where \( n_1 = j \). Since \( n_1 \) cannot change by more than \( \pm 1 \) in a single transition, all the hatched blocks of figure A1.1 are zero.

The blocks marked \( \bar{A} \) correspond to transitions where \( n_1 \) decreases by 1 due to a part completion by downstream machine \( M_1 \). Since no other event can occur during the same transition, the submatrix \( \bar{A} \) is diagonal. Half of the diagonal elements of \( \bar{A} \) are zero (those elements corresponding to states where \( \sigma_1 = 0 \)), and the other half equal \( \mu_1 \) (those corresponding to states where \( \sigma_1 = 1 \)). All blocks marked \( \bar{A} \) in the figure are identical, and they do not depend on the routing policy \( K \).

The blocks marked \( Z_{n_1}^K \) correspond to transitions where \( n_1 \) increases by 1 because the lead machine \( M_0 \) completes a part and deposits it in buffer \( B_1 \). Again, since no other event can occur during the same transition, the submatrix \( Z_{n_1}^K \) is also diagonal. Some diagonal elements of \( Z_{n_1}^K \) are zero, corresponding to states where \( M_0 \) is under repair or where the decision is to place the next part in buffer \( B_2 \) instead. All the nonzero diagonal elements \( Z_{n_1}^K \) equal \( \mu_0 \).

The diagonal blocks \( I_{n_1}^K \) of \( A^K \) have a structure very
similar to $\Delta^K$ itself, as shown in figure A1.2 for $0 < n_1 < N_1$. The matrix $Y^K_{n_1}$ is composed of $(N_2+1)^2$ submatrices, each of which has dimensions 8x8. The $(i,j)^{th}$ block of $Y^K_{n_1}$ contains the rates for transitions from states where $n_2 = i$ to states where $n_2 = j$. For reasons analogous to those given earlier, we can make the following statements about the structure of $Y^K_{n_1}$ for $0 < n_1 < N_1$:

- $Y^K_{n_1}$ is also block-tridiagonal (i.e., the hatched
  blocks in figure A1.2 are zero).

- Blocks marked $U$ are diagonal. The diagonal elements alternate between 0 and $\mu_2$. All $U$ blocks are identical, even for different $Y^K_{n_1}$ blocks, and they do not depend on the routing policy $K$.

- Blocks marked $W^K_{n_1 n_2}$ are diagonal. All nonzero diagonal elements equal $\mu_0$ and correspond to a decision to place a part in buffer $E_2$.

- Blocks marked $Y_0$, $Y$ and $Y_{N_2}$ contain the failure and repair rates. These blocks do not depend on the routing policy $K$. They are also sparse and
Figure A1.2. Structure of the Submatrices $Y_{n_1}^K$.
structured, but this will not be discussed further here. All $V$ blocks are identical, even for different $X_{n_1}^K$ blocks, so long as $0 < n_1 < N_1$.

If $n_1 = 0$ or $n_1 = N_1$, then $X_{n_1}^K$ has the same structure as described above for $0 < n_1 < N_1$, but with slightly different submatrices $V_0$, $V$ and $V_{N_2}$.

As mentioned in chapter 7, it would be interesting to adapt Howard's policy iteration algorithm (Howard [1960]) to take advantage of this structure in the transition rate matrix. Howard's algorithm starts with any routing policy $K$ and solves a system of $S$ linear equations in $S$ unknowns, where $S$ is the number of system states. The coefficient matrix for this system of equations is a perturbed version of the transition rate matrix $A^K$. The solution of these equations is used to improve the policy $K$, which changes $A^K$. Then the same system of linear equations is solved again, this time with the new $A^K$, and the solution is used to update the policy again. This process continues until an optimal policy is found. Due to the large state space, solving the system of equations even once is a formidable task. Howard's algorithm might be feasible, however, for two reasons. First, Schick and Gershwin [1978, pp. 83-111]
present an efficient algorithm for solving a system of linear equations whose coefficient matrix has the nested, block-tridiagonal structure exhibited by $A^K$. Second, if for each iteration of Howard's algorithm we update the routing decision at only one state (rather than updating the entire policy), then only two elements in one row of $A^K$ change at each iteration. The matrix inversion lemma (Householder [1965]) can therefore be used to update the solution of the system of linear equations with very little effort. (Details can be found in Schick and Gershwin [1978, p. 95].) In other words, the linear equations need only be solved once. Even so, it is not clear whether Howard's algorithm is competitive with the iterative method described in chapter 3. In particular, the memory requirements of Schick's and Gershwin's algorithm far exceed those of the iterative method.
APPENDIX 2

ANALYSIS OF PART COMPLETION TIME AT AN UNRELIABLE MACHINE

In section 4.2.2 we state that the expected time (including down time) required for an unreliable but currently operating machine to finish processing its current part is \( \frac{1}{\rho} = \frac{\lambda + \delta}{\mu r} \). In terms of the transition diagram of figure 2.1, this is the expected time until the next self-transition (marked "\( \mu \)"") occurs, given that the machine is currently in the "up" state. To simplify the following discussion, assume that the machine has only a single part available. When it completes the part, the machine passes to an artificial "finished" state. This is illustrated in figure A2.1. We would like to prove that the mean first passage time from the "up" state to the "finished" state is \( \frac{1}{\rho} \). This is shown below in two different ways. The first method is a direct first-moment computation of the mean passage time. The second method derives the entire probability density function for the first passage time, from which we compute not only the mean but also the variance and the coefficient of variation of the time to complete a part. This provides a quantitative measure of the variability of the parts flow through an unreliable machine.
Figure A2.1. Transition diagram for an unreliable machine with a single part.
Method 1*

Assume that the machine is currently in state $U$, and let $t$ be the random variable representing the first passage time from $U$ to state $F$. We can compute the expectation $E(t)$ by the law of total probability, conditioning on the destination of the first transition from state $U$.

Let us find the probability, denoted $\text{Prob}(U \rightarrow F)$, that the first transition from state $U$ is to state $F$. This is just the probability that the processing time (an exponentially distributed random variable $x$ of mean $\frac{1}{\mu}$) is less than the time until the next failure (an independent, exponentially distributed random variable $y$ of mean $\frac{1}{\rho}$). It is easy to show, by direct integration of the joint probability density function of $x$ and $y$, that the probability in question equals $\frac{\mu}{\mu + \rho}$. The complement of this quantity, $\frac{\rho}{\mu + \rho}$, is the probability that the first transition from state $U$ is to state $D$, denoted $\text{Prob}(U \rightarrow D)$. In summary:

* This proof was constructed in collaboration with Dr. John W. Palmer of Bell Laboratories, whose assistance is gratefully acknowledged.
\[ \text{Prob}(U \rightarrow F) = \frac{\mu}{\mu + p} \quad (A2.1) \]

\[ \text{Prob}(U \rightarrow D) = \frac{p}{\mu + p} \quad (A2.2) \]

Now the expectation of \( t \) **conditioned** on each of these events must be computed. First we determine the conditional expectation of the time \( t \) to get from \( U \) to \( F \), given that the first transition from \( U \) is to \( F \), which is denoted \( E(t | U \rightarrow F) \). This is the conditional expectation of the processing time (an exponentially distributed random variable \( x \) with unconditioned mean \( \frac{1}{\mu} \)), given that it is less than the time till the next failure (an independent, exponentially distributed random variable \( y \) with unconditioned mean \( \frac{1}{p} \)). Using the definition of conditional probability and the independence of \( x \) and \( y \), it is straightforward to show that the conditional probability density function for \( x \) is \((\mu + p)e^{-(\mu + p)x}\). This is again exponential, with mean \( \frac{1}{\mu + p} \). We conclude that:

\[ E(t | U \rightarrow F) = \frac{1}{\mu + p} \quad (A2.3) \]

Notice that this is symmetric in \( \mu \) and \( p \). Consequently, the conditional expectation of the time to get from \( U \) to \( D \), given that the first transition from \( U \) is to \( D \), is also \( \frac{1}{\mu + p} \):

\[ \frac{1}{\mu + p} \]
\[ E(\text{time from } U \rightarrow D \mid U \rightarrow D) = \frac{1}{\mu + \rho} \quad \text{(A2.4)} \]

(This is an example of the well-known property of Markov models that the holding time in the current state is independent of the destination state. See Howard [1971, pp. 716, 769].) If the first transition from \( U \) is to \( D \), then the random variable \( t \) is the sum of three independent quantities: the time to get from \( U \) to \( D \), the time to get from \( D \) back to \( U \), and the time to get from \( U \) to \( F \) (not necessarily in a single transition). The respective means of these three quantities are \( \frac{1}{\mu + \rho} \) (from equation (A2.4)), the mean repair time \( \frac{1}{r} \), and the unconditioned mean first passage time \( E(t) \) (since we are back where we started). In summary, the conditional expectation of the time \( t \) to get from \( U \) to \( F \), given that the first transition from \( U \) is to \( D \), is as follows:

\[ E(t \mid U \rightarrow D) = \frac{1}{\mu + \rho} + \frac{1}{r} + E(t) \quad \text{(A2.5)} \]

Using the law of total probability and substituting from equations (A2.1), (A2.2), (A2.3) and (A2.5), we can construct a simple linear equation for the unconditional expectation \( E(t) \):
\[ E(t) = E(t|U \rightarrow F) \cdot \text{Prob}(U \rightarrow F) + E(t|U \rightarrow D) \cdot \text{Prob}(U \rightarrow D) \]  

(A2.6)

\[ = \frac{1}{\mu + p} \cdot \frac{\mu}{\mu + p} + \left[ \frac{1}{\mu + p} + \frac{1}{r} + E(t) \right] \cdot \frac{p}{\mu + p} \]  

(A2.7)

Solving for \( E(t) \) yields:

\[ E(t) = \frac{p}{\mu r} = \frac{1}{\rho} \]  

(A2.8)

and the proof is complete.

**Method 2**

We would like to find the entire probability density function for the first passage time from state \( U \) to state \( F \) in figure A2.1.* If we order the three states as \( (U,D,F) \), then the transition rate matrix \( \Lambda \) for the Markov machine model is given by:

\[ \Lambda = \begin{bmatrix} -(\mu + p) & p & \mu \\ r & -r & 0 \\ 0 & 0 & 0 \end{bmatrix} \]  

(A2.9)

---

* Subsequent to the writing of this thesis, it has come to the attention of the author that the function of interest is the interarrival time distribution for the interrupted Poisson process. This distribution has been derived by Kuczura [1973].
Let us also define an interval transition probability matrix \( \hat{\Phi}(t) \). Each element \( \hat{\phi}_{ij}(t) \) gives the probability that the machine will be in state \( j \) at time \( t \), given that it was in state \( i \) at time \( 0 \). Notice that \( \hat{\phi}_{UF}(\cdot) \) is also the cumulative distribution function for the first passage time from state \( U \) to state \( F \), since \( F \) is a trapping state. Therefore, if we can first determine \( \hat{\phi}_{UF}(\cdot) \), we can then find the desired probability density function \( f(\cdot) \) for the first passage time from \( U \) to \( F \) by differentiating \( \hat{\phi}_{UF}(\cdot) \). This will be our approach.

The Laplace or exponential transform \( \hat{\phi}^e(s) \) of \( \hat{\phi}(t) \) is related to \( \Lambda \) by the following equation (Howard [1971, p. 775]):

\[
\hat{\phi}^e(s) = (sI - \Lambda)^{-1} \tag{A2.10}
\]

where \( I \) is the identity matrix. Substituting for \( \Lambda \) and performing the inversion:
\[ \Phi^e(s) = \begin{bmatrix} s+\mu+\rho & -\rho & -\mu \\ -r & s+r & 0 \\ 0 & 0 & s \end{bmatrix}^{-1} \tag{A2.11} \]

\[ = \begin{bmatrix} \frac{s+r}{s^2+s(\mu+p+r)+\mu r} & \frac{\rho}{s^2+s(\mu+p+r)+\mu r} & \frac{\mu(s+r)}{s[s^2+s(\mu+p+r)+\mu r]} \\ \frac{r}{s^2+s(\mu+p+r)+\mu r} & \frac{s+\mu+p}{s^2+s(\mu+p+r)+\mu r} & \frac{\rho \mu}{s[s^2+s(\mu+p+r)+\mu r]} \\ 0 & 0 & \frac{1}{s} \end{bmatrix} \tag{A2.12} \]

Picking off the element of interest, we have the Laplace transform of \( \phi_{UF}(t) \):

\[ \Phi^e_{UF}(s) = \frac{\mu(s+r)}{s[s^2+s(\mu+p+r)+\mu r]} \tag{A2.13} \]

Since the probability density function \( f(*) \) of the first passage time from \( U \) to \( F \) is the derivative of \( \Phi_{UF}(*) \), the Laplace transforms of these two functions are related in the following way:
\[ f^e(s) = s \phi^e_U(s) - \phi_U(t=0) \quad (A2.14) \]
\[ = s \phi^e_U(s) - 0 \quad (A2.15) \]
\[ = \frac{\mu(s+r)}{s^2 + s(\mu + p + r) + \mu r} \quad (A2.16) \]

Inverting this transform gives the desired probability density function:

\[ f(t) = \mu \left[ ce^{-at} + (1-c)e^{-bt} \right] \quad (A2.17) \]

where

\[ a = \frac{1}{2} \left[ \mu + p + r + \sqrt{(\mu + p + r)^2 - 4r\mu} \right] \quad (A2.18) \]
\[ b = \frac{1}{2} \left[ \mu + p + r - \sqrt{(\mu + p + r)^2 - 4r\mu} \right] \quad (A2.19) \]
\[ c = \frac{1}{2} \left[ 1 + \frac{\mu + p - r}{\sqrt{(\mu + p + r)^2 - 4r\mu}} \right] \quad (A2.20) \]

Thus the first passage time from U to F, which represents the time until a currently operating machine completes its current part, has a hyperexponential distribution (Kleinrock [1975, p. 141]); that is, the distribution is a convex or probabilistic combination of the two exponential distributions \(ae^{-at}\) and \(be^{-bt}\).
The mean and variance of the time to complete a part can be computed directly from (A2.17), or the moment-generating properties or the Laplace transform (A2.16) can be exploited. These moments along with the coefficient of variation (i.e., the ratio of the standard deviation to the mean) are given below:

$$\text{Mean} = \frac{x + p}{\mu r} = \frac{1}{\rho} \quad (A2.21)$$

$$\text{Variance} = \frac{(x + p)^2 + 2px}{\mu^2 r^2} = \frac{1}{\rho^2} + \frac{2p}{\mu r^2} \quad (A2.22)$$

$$(\text{Coefficient of variation})^2 = 1 + \frac{2pu}{(x + p)^2} \quad (A2.23)$$

The variance (especially the second term of (A2.22)) and the coefficient of variation can be rather large. For instance, consider a machine with mean processing time $\frac{1}{\mu} = 1$, mean time between failures $\frac{1}{p} = 100$, and mean repair time $\frac{1}{r} = 25$. The isolated efficiency $e$ of the machine is 80%, and its isolated production rate $\rho$ is 0.8. The mean time to complete a part is $\frac{1}{\rho}$ or 1.25, with a variance of 14.1 and a coefficient of variation of 3. By comparison, the variance for a perfectly reliable machine with the same isolated production rate is $\frac{1}{\rho^2}$ or 1.56, and the coefficient of variation is 1.
Note from equation (A2.21) that the mean time spent by a part in the machine depends on the failure rate \( p \) and the repair rate \( r \) only through their ratio \( \frac{p}{r} \). This is not true for the variance and coefficient of variation, however. A machine with a small repair rate and an extremely small failure rate (compared to its processing rate \( \mu \)) would have excellent performance as measured by the mean time per part (A2.21), but very poor performance by the second-moment measures (A2.22) and (A2.23). The parts flow through such a machine is so variable that if the machine were placed in a manufacturing network, very large buffers would be required to smooth its production and prevent starvation and blockage in the system. We conclude that first moment measures of isolated machine reliability and performance are not sufficient to gauge the machine's effectiveness in the larger network.

Finally, we note that the variance and the coefficient of variation of the part completion time are more sensitive to the repair rate \( r \) than to the failure rate \( p \). One way to see this is to compute and compare the relative derivatives \( \frac{\Delta x}{\Delta r} \) and \( \frac{\Delta x}{\Delta p} \), where \( x \) can be either the variance or the square of the coefficient of variation. These relative derivatives indicate the per-
centage change in $x$ corresponding to a percentage change in $r$ or $p$, respectively. It is easy to show that $\frac{dx}{dr}$

is greater in magnitude than $\frac{dx}{dp}$. 
APPENDIX 3

THE SHORT-TERM ROUTING STRATEGY

Here we derive a formula for the short-term routing strategy of section 5.2.2, applied to our manufacturing system model with three perfectly reliable machines. This strategy routes each workpiece so as to maximize the expected short-term production over the time it will take the lead machine to make another part. This routing rule is useful for approximating the intercept of the optimal decision boundary, as discussed in section 5.2.2.

First we state some useful facts concerning exponentially distributed random variables, along with sketches of the proofs.

Proposition A3.1. Let $x_0$ be an exponentially distributed random variable with mean $\frac{1}{\mu_0}$. Let $x_1, x_2, \ldots, x_n$ be exponentially distributed random variables, each with mean $\frac{1}{\mu}$ ($n \geq 1$). All the random variables are independent. Then the probability that the sum of $x_1$ through $x_n$ is less than $x_0$ is given by:
\[ \text{Prob} \left( x_1 + x_2 + \cdots + x_n < x_0 \right) = \left[ \frac{\mu}{\mu + \mu_0} \right]^n \] (A3.1)

**Proof:** There are two ways to prove this proposition. One is to recognize that the sum \( s \) of \( x_1 \) through \( x_n \) has an Erlang distribution of order \( n \) with mean \( \frac{n}{\mu} \) and that \( s \) and \( x_0 \) are independent. We can easily integrate the joint probability density function of \( s \) and \( x_0 \) over that half of the \( (s, x_0) \) plane where \( s < x_0 \).

The other proof is by induction. The case for \( n = 1 \) is proved by integrating the joint probability density function of \( x_1 \) and \( x_0 \) over that half of the \( (x_1, x_0) \) plane where \( x_1 < x_0 \). The induction step uses the Markov or memoryless property of the exponential distribution of \( x_0 \) (Feller [1971, pp. 8-9]) to show that:

\[
\text{Prob}(x_1 + x_2 + \cdots + x_n < x_0)
= \text{Prob}(x_n < x_0) \cdot \text{Prob}(x_1 + x_2 + \cdots + x_n < x_0 \mid x_n < x_0) \]
\[
= \text{Prob}(x_n < x_0) \cdot \text{Prob}(x_1 + x_2 + \cdots + x_{n-1} < x_0) \quad (A3.2)
\]
\[
= \frac{\mu}{\mu + \mu_0} \cdot \text{Prob}(x_1 + x_2 + \cdots + x_{n-1} < x_0) \quad (A3.3)
\]
\[
= \frac{\mu}{\mu + \mu_0} \cdot \text{Prob}(x_1 + x_2 + \cdots + x_{n-1} < x_0) \quad (A3.4)
\]
Proposition A3.2. Let \( x_0 \) be an exponentially distributed random variable with mean \( \frac{1}{\mu_0} \). Let \( x_1, x_2, \ldots, x_m, x_{m+1} \) be exponentially distributed random variables, each with mean \( \frac{1}{\mu} \) (\( m \geq 1 \)). All the random variables are independent. Then the probability that \( x_0 \) is between the sum of \( x_1 \) through \( x_m \) and the sum of \( x_1 \) through \( x_{m+1} \) is given by:

\[
\text{Prob} \left( x_1 + x_2 + \ldots + x_m < x_0 < x_1 + x_2 + \ldots + x_m + x_{m+1} \right) = \left[ \frac{\mu_0}{\mu + \mu_0} \right]^m \cdot \left[ \frac{\mu}{\mu + \mu_0} \right]^m
\]  

(A3.5)

Proof: Again, there are two ways to prove this proposition. One is to recognize that the sum \( s \) of \( x_1 \) through \( x_m \) has an Erlang distribution of order \( m \) with mean \( \frac{m}{\mu} \) and that \( s, x_{m+1}, \) and \( x_0 \) are all independent. We can easily integrate the joint probability density function of \( s, x_{m+1}, \) and \( x_0 \) over that region of \( (s, x_{m+1}, x_0) \) space where \( s < x_0 < s + x_{m+1} \).

The other proof exploits the memoryless property of the exponential distribution of \( x_0 \) to show that:
\[
\text{Prob}(s < x_0 < s + x_{m+1})
\]
\[
= \text{Prob}(s < x_0) \cdot \text{Prob}(x_0 < s + x_{m+1} | s < x_0) \quad (A3.6)
\]
\[
= \text{Prob}(s < x_0) \cdot \text{Prob}(x_0 < x_{m+1}) \quad (A3.7)
\]

Applying proposition A3.1 to each of the two factors of equation (A3.7) yields the desired result (A3.5).

We prove one more fact which will be helpful later.

**Proposition A3.3.**

\[
\sum_{m=1}^{n-1} ma^m = \frac{(n-1)a^{n+1} - na^n + a}{(1-a)^2} \quad (A3.8)
\]

**Proof:**

\[
\sum_{m=1}^{n-1} ma^m = a \cdot \sum_{m=1}^{n-1} ma^{m-1} \quad (A3.9)
\]
\[
= a \cdot \frac{d}{da} \left[ \sum_{m=1}^{n-1} a^m \right] \quad (A3.10)
\]
\[
= a \cdot \frac{d}{da} \left[ \frac{a^n}{1-a} \right] \quad (A3.11)
\]
\[
= a \cdot \frac{(n-1)a^n - na^{n-1} + 1}{(1-a)^2} \quad (A3.12)
\]
\[
= \frac{(n-1)a^{n+1} - na^n + a}{(1-a)^2} \quad (A3.13)
\]
Now we are ready to construct the short-term routing rule. Consider two perfectly reliable machines $M$ and $M_0$. Processing times for these machines are exponentially distributed, with mean $\frac{1}{\mu}$ for $M$ and $\frac{1}{\mu_0}$ for $M_0$. Machine $M$ has $n$ available workpieces, while $M_0$ has only one. Let the random variable $m$ denote the number of parts $M$ completes in the time $M_0$ processes its single part. We would like to determine the expected value of $m$, which we shall call $D(n; \mu, \mu_0)$. The probability distribution $f(\cdot)$ for $m$ is easy to find. By proposition A3.1 we have:

$$f(0) = \frac{\mu_0}{\mu + \mu_0}$$  \hspace{1cm} (A3.14)

and

$$f(n) = \left[\frac{\mu}{\mu + \mu_0}\right]^n$$  \hspace{1cm} (A3.15)

By proposition A3.2 we have:

$$f(m) = \left[\frac{\mu_0}{\mu + \mu_0}\right] \cdot \left[\frac{\mu}{\mu + \mu_0}\right]^m \text{ for } 0 < m < n$$  \hspace{1cm} (A3.16)

The expected value of $m$ is now easily computed:
\[ D(n; \mu, \mu_0) = \sum_{m=0}^{n} m \cdot f(m) \]

\[ = 0 \cdot \left[ \frac{\mu_0}{\mu + \mu_0} \right] + \sum_{m=1}^{n-1} m \cdot \left[ \frac{\mu_0}{\mu + \mu_0} \right] \cdot \left[ \frac{\mu}{\mu + \mu_0} \right]^m + n \cdot \left[ \frac{\mu}{\mu + \mu_0} \right]^n \]

(A3.17)

\[ = \left[ \frac{\mu_0}{\mu + \mu_0} \right] \cdot \sum_{m=1}^{n-1} m \cdot \left[ \frac{\mu}{\mu + \mu_0} \right]^m + n \cdot \left[ \frac{\mu}{\mu + \mu_0} \right]^n \]

(A3.18)

By proposition A3.3 and some algebraic manipulation, this reduces to:

\[ D(n; \mu, \mu_0) = \frac{\mu}{\mu_0} \left[ 1 - \left[ \frac{\mu}{\mu + \mu_0} \right]^n \right] \]

(A3.19)

The short-term strategy routes the current part so as to maximize the expected number of parts produced by the downstream machines during the time it will take the lead machine to produce its next part. Let \( n_1 \) and \( n_2 \) denote the buffer levels at the time a part is completed at the lead machine. If that part is routed to buffer \( B_1 \), then the expected short-term production will be
\[ D(n_1+1; \mu_1, \mu_0) + D(n_2; \mu_2, \mu_0) \]. If the current part is routed to \( B_2 \), then the expected production is \[ D(n_1; \mu_1, \mu_0) + D(n_2+1; \mu_2, \mu_0) \]. Thus the short-term routing decision is 1 if:

\[ D(n_1+1; \mu_1, \mu_0) + D(n_2; \mu_2, \mu_0) > D(n_1; \mu_1, \mu_0) + D(n_2+1; \mu_2, \mu_0) \]  

(A3.20)

and is 2 otherwise. Substituting formula (A3.19) for \( D \) and simplifying, we find that the short-term routing decision is 1 if and only if:

\[ \left( \frac{\mu_1}{\mu_1 + \mu_0} \right)^{n_1+1} > \left( \frac{\mu_2}{\mu_2 + \mu_0} \right)^{n_2+1} \]  

(A3.21)

Taking the negative logarithm of both sides, this becomes:

\[ (n_1+1) \ln \left[ 1 + \frac{\mu_0}{\mu_1} \right] < (n_2+1) \ln \left[ 1 + \frac{\mu_0}{\mu_2} \right] \]  

(A3.22)

Thus the short-term decision boundary is described by the equation:
\[
\frac{n_2 + 1}{n_1 + 1} = \frac{\ln \left[ \frac{\mu_0}{\mu_1} \right]}{\ln \left[ \frac{\mu_0}{\mu_2} \right]} \quad (A3.23)
\]

This is a straight line of slope

\[
\frac{\ln \left[ \frac{\mu_0}{\mu_1} \right]}{\ln \left[ \frac{\mu_0}{\mu_2} \right]} \quad (A3.24)
\]

and intercept

\[
\frac{\ln \left[ 1 + \frac{\mu_0}{\min(\mu_1, \mu_2)} \right]}{\ln \left[ 1 + \frac{\mu_0}{\max(\mu_1, \mu_2)} \right]} - 1 \quad (A3.25)
\]

This intercept is a very good approximation to the intercept of the optimal decision boundary.

Neither the slope nor the intercept of this boundary match those of the other closed-loop heuristics. For comparison, all three heuristics and the optimal policy are plotted in figure A3.1 for a system with \( \mu_0 = 40, \mu_1 = 5, \mu_2 = 35, N_1 = 9, \) and \( N_2 = 49. \) (This system is balanced but very asymmetric, so the \( n_1 \) and \( n_2 \)
Figure A3.1. Optimal strategy and three feedback heuristics.
axes of figure A3.1 are scaled differently. For reference, we include the line where \( n_1 = n_2 \).
APPENDIX 4

A ROUTING STRATEGY BASED ON STARVATION PROBABILITIES

In sections 4.1 and 7.1, a closed-loop routing strategy is suggested which is similar in spirit to the worktime balance heuristic. This strategy routes a part to the downstream machine which would be likely to starve first if the lead machine were to stop processing parts. We can easily derive formulas for this rule for the case of perfectly reliable downstream machines. Let $x_1$ be a random variable denoting the time till starvation of machine $M_1$, and let $x_2$ be the corresponding random variable for machine $M_2$. Then we can express the routing rule in three equivalent ways:

Route to: $M_1$ if $\text{Prob}(x_1 < x_2) > \frac{1}{2}$ \hspace{1cm} (A4.1)

$M_2$ if $\text{Prob}(x_1 < x_2) < \frac{1}{2}$

or

Route to: $M_1$ if $\text{Prob}(x_2 < x_1) < \frac{1}{2}$ \hspace{1cm} (A4.2)

$M_2$ if $\text{Prob}(x_2 < x_1) > \frac{1}{2}$

or

Route to: $M_1$ if $\text{Prob}(x_1 < x_2) > \text{Prob}(x_2 < x_1)$ \hspace{1cm} (A4.3)

$M_2$ if $\text{Prob}(x_1 < x_2) < \text{Prob}(x_2 < x_1)$

It is straightforward to compute the probability
that $x_1 < x_2$, since $x_1$ and $x_2$ are independent, $x_1$ has an Erlang distribution of order $n_1$ with mean $\frac{n_1}{\mu_1}$, and $x_2$ has an Erlang distribution of order $n_2$ with mean $\frac{n_2}{\mu_2}$. (Since this strategy always routes a part to a starving machine, we assume for this discussion that $n_1 > 0$ and $n_2 > 0$.) For reference, we give the probability density function $f(x;n,\mu)$ and the cumulative distribution function $F(x;n,\mu)$ for an Erlang distribution of order $n$ with mean $\frac{n}{\mu}$ (Feller [1971], p. 11):

$$f(x;n,\mu) = \frac{n x^{n-1} e^{-\mu x}}{(n-1)!} \quad (A4.4)$$

$$F(x;n,\mu) = 1 - e^{-\mu x} \sum_{k=0}^{n-1} \frac{(\mu x)^k}{k!} \quad (A4.5)$$

The probability that $x_1 < x_2$ can be expressed in terms of $f$ and $F$:

$$\text{Prob}(x_1 < x_2) = \int_0^\infty \text{Prob}(x < x_2) \cdot f(x; n_1, \mu_1) \, dx \quad (A4.6)$$

$$= \int_0^\infty [1 - \text{Prob}(x_2 < x)] \cdot f(x; n_1, \mu_1) \, dx \quad (A4.7)$$

$$= \int_0^\infty [1 - F(x; n_2, \mu_2)] \cdot f(x; n_1, \mu_1) \, dx \quad (A4.8)$$

If we substitute formulas (A4.4) and (A4.5) into (A4.8), exchange the order of integration and summation, and simplify, we arrive at the desired result:
\[
\text{Prob}(x_1 < x_2) = \left[ \frac{\mu_1}{\mu_1 + \mu_2} \right]^{n_1} \sum_{k=0}^{n_2-1} \left[ \frac{\mu_2}{\mu_1 + \mu_2} \right]^k \frac{(k+n_1-1)!}{k!(n_1-1)!} \quad (A4.9)
\]

(The ratio of factorials is the familiar formula for the number of combinations of \(k+n_1-1\) objects taken \(k\) at a time.) By symmetry, then:

\[
\text{Prob}(x_2 < x_1) = \left[ \frac{\mu_2}{\mu_1 + \mu_2} \right]^{n_2} \sum_{k=0}^{n_1-1} \left[ \frac{\mu_1}{\mu_1 + \mu_2} \right]^k \frac{(k+n_2-1)!}{k!(n_2-1)!} \quad (A4.10)
\]

In summary, our routing rule as given in (A4.1) compares the value of expression (A4.9) with the quantity \(\frac{1}{2}\) and routes accordingly. If \(n_1 < n_2\), it is easier to compute expression (A4.10), so the routing policy as given in (A4.2) uses (A4.10) in its comparison. Version (A4.3) of the routing law compares both expressions (A4.9) and (A4.10), allowing us to simplify the formula somewhat. The resulting rule routes a part to \(M_1\) if:

\[
\sum_{k=1}^{n_2} \left[ 1 + \frac{\mu_1}{\mu_2} \right]^k \frac{(n_2-k)!}{(n_2-k)!} > \sum_{k=1}^{n_1} \left[ 1 + \frac{\mu_2}{\mu_1} \right]^k \frac{(n_1-k)!}{(n_1-k)!} \quad (A4.11)
\]

and otherwise routes a part to \(M_2\).
It would be interesting to graph this strategy to see how it compares with the worktime balance heuristic and the optimal policy.
REFERENCES


Berman, O. [1979], "Efficiency and Productivity Rate of a Transfer Line with Two Machines and a Finite Storage Buffer," Report No. LIDS-R-899, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, April 1979.


Buzen, J. [1971 a], "Analysis of System Bottlenecks Using a Queueing Network Model," *ACM SIGOPS Workshop on System Performance Evaluation,* Harvard University, pp. 82-103, April 1971.


Chen, P. P.-S. [1973 a], "Optimal Partition of Input Load to Parallel Exponential Servers," *Proceedings


Rolfe, A. J. [1968], "The Control of a Multiple Facility, Multiple Channel Queueing System with Parallel Input Streams," Technical Report No. 22, Graduate School of Business, Stanford University, March 1968.


