Scheduling Flexible Machining Systems when Machines are Prone to Failure

by

Richard R. Hildebrant

B.S., State University of New York at Buffalo, 1970

S.M., Massachusetts Institute of Technology, 1972

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1980

Signature of Author ----------
Department of Aeronautics and Astronautics
May 29, 1980

Certified by ----------
Thesis Supervisor

Certified by ----------
Thesis Supervisor

Certified by ----------
Thesis Supervisor

Accepted by ----------
Chairman, Department Graduate Committee

SCHEDULING FLEXIBLE MACHINING SYSTEMS

WHEN

MACHINES ARE PRONE TO FAILURE

by

Richard R. Hildebrant

Submitted to the Department of Aeronautics and Astronautics on May 29, 1980, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

A Flexible Machining System (FMS) is aptly described as an automated job-shop. Machines are connected to one another by an automated material transporter and the system is intended to process a number of different kinds of parts. The system incorporates redundancy of function which allows the processing of parts even though machine failures may have occured. This research develops a method for scheduling parts that considers and ,in fact, anticipates machine failure.

Motivated by the complexity of the problem, a structure for decision making is developed that identifies the fundamental decisions, and then systematically simplifies their determination. A multi-level framework results where global, low resolution models (network queueing models coupled with optimization algorithms), are integrated with local, high resolution models (deterministc scheduling). In standard hierarchical manner, the decisions of the upper levels are accepted by the lower levels as constraints. Scheduling algorithms are developed, implemented and demonstrated on representative machining systems. Depending on the system and part set, substantial improvement over more naive strategies can be realized.

Thesis Supervisor: Sanjoy K. Mitter, Ph.D.
Title: Proffessor of Electrical Engineering, MIT

Thesis Supervisor: Rajan Suri, Ph.D.
Title: Assistant Proffessor of Division of Applied Sciences, Harvard

Thesis Supervisor: Thomas L. Magnanti, Ph.D.
Title: Proffessor of Operations Research and Management, MIT

ACKNOWLEDGMENTS

It would have been impossible to complete this degree and thesis without the technical, financial and moral suport of others. It is appropriate to mention their contribution.

I am indebted to the chairman of my advisory committee, Sanjoy Mitter, for providing his support and encouragement at every stage of the program. Also, my contacts with Thomas Magnanti were very rewarding. I am especially grateful to Rajan Suri whose interest and advice provided intellectual stimulus to me and impetus to the work when it was needed most.

Thanks are due to my colleagues at C.S.Draper Laboratory, Digital Computation Division, for providing the necessary funding and encouragement which allowed me to carry out the research that led to this thesis. In addition, Rita Connolly and her staff, Tara Lee and Annie DiRenzo did an excellent and timely job of typing this document.

Finally, my warmest appreciation goes to Vicki Schwartz for her understanding and companionship during the course of this thesis.

TABLE OF CONTENTS

Notation

The following notation conventions are used in this thesis:

1. $\vec{x}$ = Vector quantity.

2. If J is a set of elements,
   then $|J|$ = number of elements in set J.

The following equivalences are used within the text:

1. $\vec{x}$ = vect(x).

2. x_i = x sub(i).

CHAPTER 1

INTRODUCTION TO FLEXIBLE MACHINING SYSTEMS

1.1 Introduction

This thesis is concerned with a new scheduling problem. The problem is associated with a system that has recently emerged from within the machining industry - the Flexible Machining System (FMS). An FMS consists of a set of machines in which parts are automatically transported under computer control from one machine to another for processing. The system is generally designed to process a number of different kinds of parts and is intended to operate without manual intervention. One might consider an FMS to be an automated job-shop, although it is quite a bit more complex as we shall see.

The significance of the scheduling problem for these systems springs from their high initial cost and the high interaction that occurs among the system's various elements. An FMS user cannot afford low system utilization, but the means for increasing utilization are often not apparent. This is particularly true if one considers that machines are apt to fail. A system of ten machines, each one of which is down ten to thirty percent of the time, is likely to harbor at least one broken machine. This could severely inhibit production, except that, FMS's are often structured so that any one of several machines can process a given part. Redundancy of function significantly raises system reliability but not without its price. The complexity of the scheduling problem is substantially increased along with system flexibility. Not only must parts be optimally directed through a system of high complexity, but the configuration of the system is continually changing as well.

Two overriding concerns have shaped the course of this research. First, we believe the scheduling problem deserves a global treatment. The sole use of techniques that admit a "short" view of the problem is shunned. Secondly, we recognize the constraint on computation imposed by the availabe computer. The scheduling concepts and algorithms proposed in this thesis are designed to adhere to this limitation.

1.2 General Flexible Machining System Description

1.2.1 Motivation for FMS's

The machining industry has, in the past, efficiently processed two classes of parts, high-volume parts and low-volume parts. Component parts associated with mass produced end-items such as automobiles are typical examples included in the high-volume catagory. In this case, it is economically advantageous to build a dedicated machining system called a transfer line for each individual variety of part.

A transfer line is an arrangement of machines in which parts are automatically transferred from one machine to another and each machine performs a certain set of operations (drilling, boring, etc.) in parallel. Of course, to gang operations together and perform them simultaneously is the key to the system's high productivity and is the chief reason for its economy. Multiple-spindle tooling is generally integral to the machine (for all practical purposes), so the system cannot be converted to process different parts. The system is extremely inflexible.

Low-volume parts, arising from products whose production requirements are small, say fewer than 50, would include prototype parts and other special needs. The system that processes the low-volume part need not necessarily do so at a high rate but must be able to process different kinds of parts without incurring a high change-over cost. It must be flexible. Standard manual job-shops as well as stand-alone, Numerical Control (N/C) machines (these are single-spindle machines that can easily be programmed to automatically perform a sequence of operations on a part) are machining alternatives for the low-volume part.

A large percentage of the machining that is being done presently is considered to lie in the medium-volume category. Agricultural and defense equipment (tractors, tanks, etc.) are two areas that contribute to this group. Machining these parts poses problems because their production requirements are either too low or too high to be economically processed on one of the two machining systems previously mentioned.

The central reason for designing Flexible Machining Systems (FMS) is to economically process medium-volume parts. The design philosophy has been to combine the high-volume features of transfer lines (multiple-spindle heads, automatic transfer of parts among machines) with the flexibility of stand-alone equipment. A number of medium-volume parts can then be grouped and processed on one system, and no single part need bear the system's high fixed cost.

1.2.2 System Configuration

The general concept of an FMS allows tremendous freedom in the way a system is configured and what its capabilities are. Each FMS is likely to incorporate features that distinguish it from all others so no two systems will be exactly the same. This thesis will not treat the most general system, but to give the reader some perspective and to introduce some of the terminology that will be used, we show a general FMS in Figure 1.1. This FMS is only conceptual, but it includes most resources that have been found to date in systems of this type. A description of each resource is given next:

Machines — The system is composed of a number of machines that may or may not be identical. A machine performs an operation on a part and so may be generalized to include inspection stations, washing stations, loading/unloading stations, etc.

Material Handling System (MHS) — Parts are transported from one machine to another with this resource and can take many forms. Figure 1.1 shows a conveyer MHS. Very often, a number of individual carts are employed that pick up parts from one machine and transport them to another. The MHS often serves double-duty as an in-process storage device for parts waiting to be machined.

Pallet/ Fixture (pixture) — Each part must be attached to a fixture for easy handling internal to the system and for proper presentation to the machines. Because of their different shapes and sizes, a fixture is tailored to each variety of part and one kind of part generally cannot use those fixtures designed for another. There are cases, though, where this is not true. Different parts may share a fixture, either concurrently (more than one part on a fixture) or separately. An important aspect of the problem is that a limited number of fixtures exist for each part. It will be seen that these constraints can affect part scheduling in subtle ways. In practice fixtures are generally not separated from pallets. For this reason, we call the combination a *pixture*.

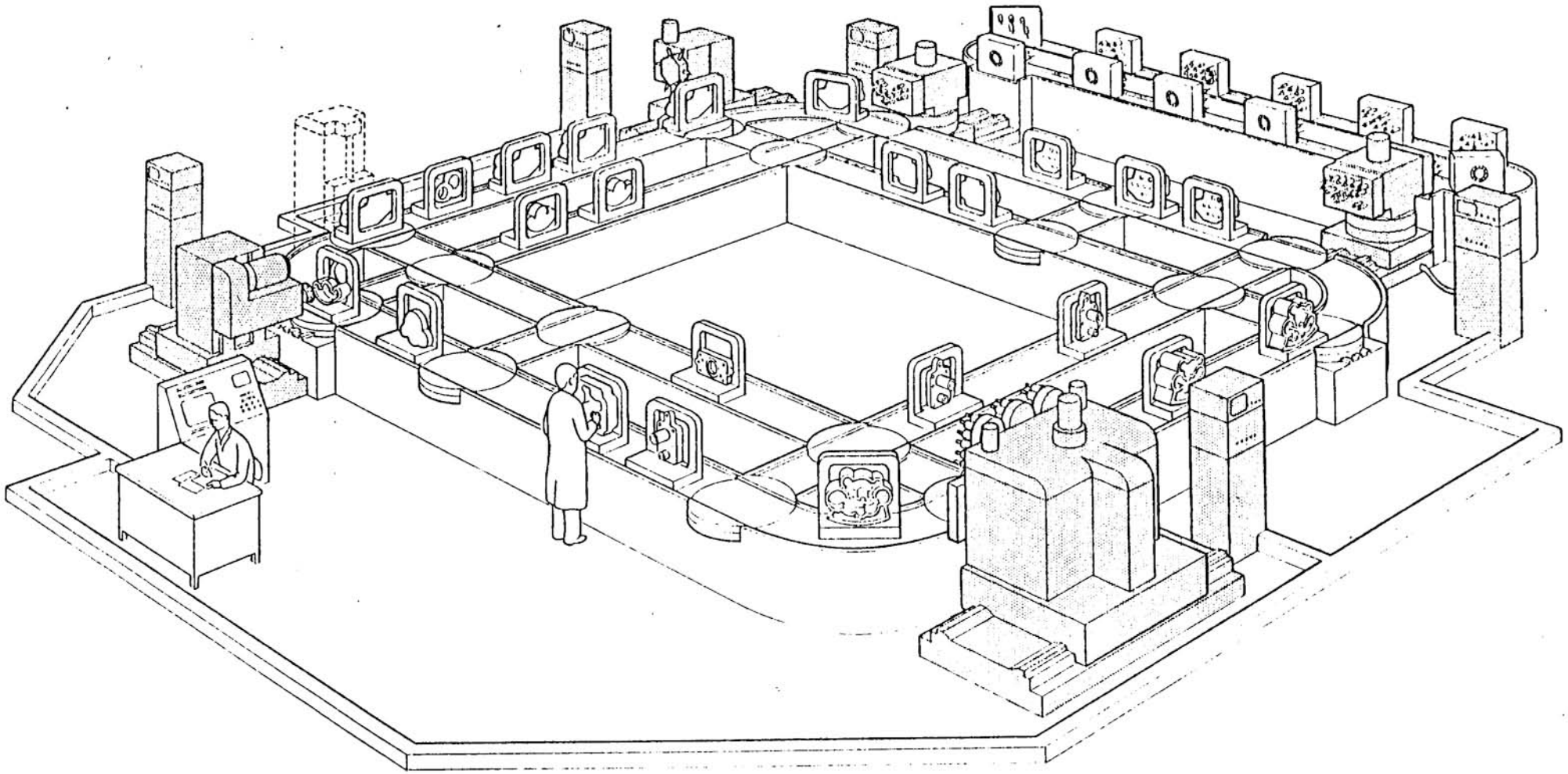


Figure 1.1: Conceptual FMS

Machine Buffer Storage

A dedicated part storage area is provided in many systems at the input and output of each machine. (Figure 1.1 does not clearly illustrate this feature.) Other systems provide a common storage area. Buffers help to isolate each machine from fluctuations in its supply of parts; they act as mini-inventories.

Tools

Tools might be considered part of the machine and, therefore, not a resource in their own right. When tools are shared among machines, this is no longer the case. Bringing parts together with tools on one machine then becomes a compound scheduling problem; both parts and tools must be scheduled.

Computer

The analyst concerned with scheduling parts through the system recognizes the computer as a resource with inherent limitations in its capacity. The computer controls much, if not all, of the decision making for the system. Parts are automatically routed through the system, machines are told how and when to process parts, all with the help of the computer.

1.2.3 Part Set

The manner in which parts are attached to their fixtures and the limited articulation of the machine tools, generally prevent a part from being completely processed on one pass through the system. It is not unusual for a part to require three to six refixturings before it is completely processed. These individual part "sides" are termed setups because they must be setup on a fixture before entering the system. From a scheduling point of view, each setup is a significant and distinct "job" and must be considered as such. It is not the number of parts that determines the size of the scheduling problem, it is the number of setups.

Precedence relations generally constrain the order in which the setups of a particular part can be processed. The need to perform "rough" before "finish" machining and the need to cut reference surfaces first are two reasons for these constraints. Figure 1.2 shows an example precedence diagram where Setup 1 and Setup 2 must both be processed before Setup 3, but in no particular order.

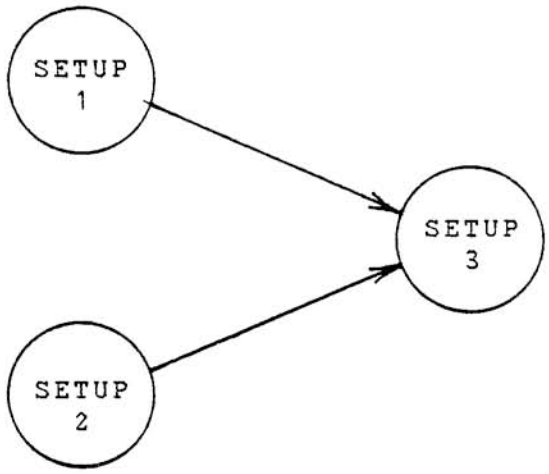


Figure 1.2 Setup Precedence Diagram

Each setup visits one or more machines on its journey through the system and at each machine an operation is performed. A machine visit may require a number of elementary operations to be performed (individual hole drillings, etc.) but it is the overall time spent at a machine that is significant for scheduling setups. This machine visit time is called the operation processing time. Operations need not be performed in a strict order; partial precedence relations are quite possible. An example operation precedence diagram is shown in Figure 1.3, illustrating another important aspect of the problem. Tooling may exist on different machines for some, if not all, operations. This is normally done as insurance against the event of machine failure; if a machine fails, parts can still be machined on alternative machines.

Since each of the four operations can be processed on one of two machines, there are, in this case, $2^4 = 16$ possible operation/machine allocations. Each of these sixteen alternatives is called a routing.

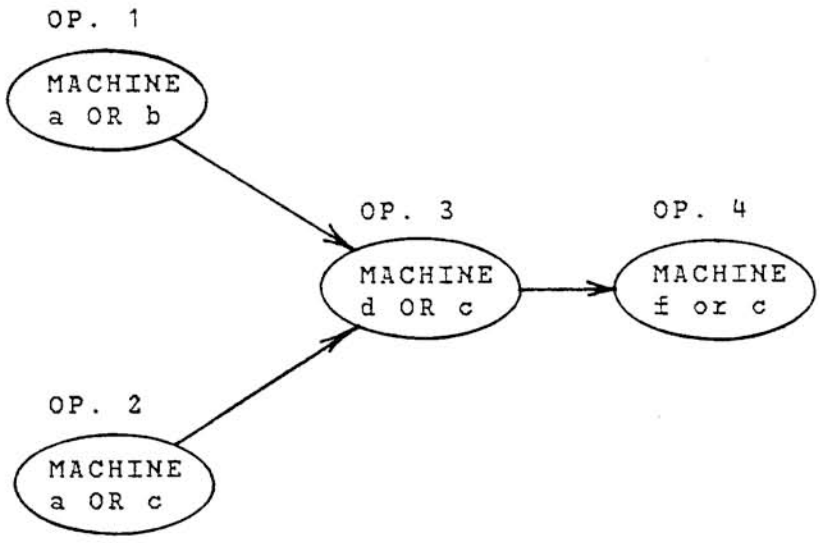


Figure 1.3 Operation Precedence Diagram

1.2.4 A Simplified FMS

The general FMS described above is an extremely complex system. Embracing all of its complicating elements at once does not seem analytically feasible, so simplifications will have to be made. But where? Ignoring certain aspects of the problem may completely miss the mark and render any methodology developed inapplicable. We wish to analyze an FMS that balances tractability against applicability, one that embodies the important elements common to a large class of systems that have been and will be built.

The fundamental FMS characteristics considered in this thesis are listed below:

System Configuration

1. The number of machines in the system is arbitrary.

2. No blocking effects due to congestion on the MHS are felt.

   A setup is blocked when it cannot proceed to its destination due to the presence of another setup. However, the effect of this blocking may not be felt! If a setup's destination machine has sufficient work to keep it busy until the setup arrives, there is no real delay in its production. So although blocking may indeed occur, the effects of blocking (as perceived through production rate) may not.

3. No blocking occurs at machines.

   The MHS may not have the capacity to retrieve parts from machines as soon as an operation is completed. If another part is waiting at the machine's input buffer, it is said to be blocked. Sufficient space in the machine's output buffer however, can eliminate this problem.

4. Sufficient queueing space exists internal to the system, either dedicated storage or common storage.

   We do not mean to imply that "infinite" queues should be possible. Our treatment of the system is such that the number of jobs internal to the system never exceeds some maximum value on the order of 15.

5. Tooling is not shared among machines.

   Although shared tooling has the potential of providing great flexibility at low cost, dedicated tooling is by far the norm on systems in use today.

6. Each setup requires a unique pixture and upper bounds on the number of each type of pixture exist.

Part Set

7. Partial precedence relations for operations are allowed.

8. Machine alternatives for each operation are allowed.

9. Precedence relations among setups are not allowed.

   Operationally, precedence constraints among setups can be freed by building up a suitable inventory of each setup. Consider two setups of a part with the following precedence diagram.

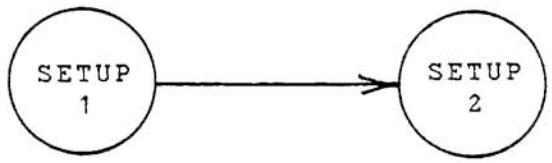


Figure 1.4

They may be isolated from one another if at some prior time, several Setup 1 pieces are processed before any setup Setup 2 pieces are done.

System Disturbances

10. Machines may fail in random manner.

Two important system characteristics not found in other treatments of the FMS problem are included:

i. Constraints on pixture availability exist (#6).

ii. Machines are assumed to fail (#11).

1.3 Statement of the Problem

A formal statement of the problem requires specification of a model and an objective.

1.3.1 System Model

Following the terminology of Coffman (1976), the system model is comprised of two parts:

(i) Resources. The system contains a set of processors M= $\{m_1, m_2, \ldots,\}$. The processors we have in mind are the machines and the MHS. Due to machine failure, this set is not constant; we shall assume a probabalistic model for future processor availability. In addition, a task may require another resource during its entire execution on some processor. The pixtures fall into this category and are described by: F = $\{F_1, F_2, \ldots\}$ where F sub(i) is the set of pixtures of type i.

(ii) Task System. A task system is described by the 5-tuple (ψ, <, {M sub(i)}, {f sub(i)}, [t(ij)]) as follows:

ψ	= $\{\xi_1, \xi_2, \ldots\}$ is a set of tasks to be performed. Each setup is generally comprised of a number of individual tasks (machine operations, MHS movements, etc.)
<	is a partial order defined on ψ that specifies the operation precedence constraints.
Each Msub(i)	is a set of processors, one of which must be chosen for task ξ sub(i). Alternate machines for operations would be expressed through M sub(i).
Each fsub(i)	is an index for the pixture type used by task i. One member of F sub(f sub(i)) is chosen for each task i.

[t(ij)]	is a matrix of execution times where t(ij) is the time to execute ξ sub(i) on processor m sub(j). These times are assumed deterministic.

1.3.2 System Objective

The system objective must accurately reflect the will of management. Management generally specifies a production target that it hopes to attain over a certain time horizon. For example, it might desire to produce 50 of Part 1, 90 of Part 2, etc., over a period of a week. At the operational level, however, this goal often translates to: "finish the production target as soon as possible." This later interpretation of the original objective is the one this thesis considers and can be simply stated as:

Minimize T

where

T = completion time of the production target

However, since the system model includes machine failure, which can only be described in a stochastic manner, we will find it convenient to minimize the expected value of T.

1.3.3 System Control Variables

The above statement of the problem is complete, although a bit austere. In order to provide a better feel for what must be done, we may state, in general terms, what the controls for the system are:

1. Sequence of setups into the system.
2. Loading Time for each setup in the sequence.
3. Machine Assignment for each operation of each setup.
4. Order of Processing of Operations for each setup.

The difficulty of the problem stems from the combinatorial nature of the controls and the number of different combinations possible.

FMS's with intricate material handling systems may have additional controls. Examples include: determining the path to transport setups from machine to machine and choosing an appropriate cart when carts are employed for transport. However, consideration of MHS decisions is beyond the scope of this thesis. We assume that no controls are associated with the MHS (this is the case with simple loop conveyor MHS'S), or that sufficient capacity exists on the MHS to make decision-making trivial.

1.4 Previous Research

Two rather large bodies of research seem directly applicable to the FMS scheduling problem, deterministic scheduling and queueing theory. Briefly, the most relevant facts about each area are discussed below.

Deterministic Scheduling

Deterministic scheduling procedures are concerned with finding the ordering of jobs on servers such that some objective is met. The last 25-30 years have witnessed a tremendous amount of research covering most circumstances [Conway et al, 1967; Baker, 1974]. One of the most studied problems in this area corresponds somewhat to the FMS problem. It is the job-shop scheduling problem.

A fundamental characteristic of the FMS, and one that is common to job-shops as well, is the manner in which parts proceed through the system. In the standard job-shop formulation, partial precedence relations and alternate machines are not allowed, but different parts can visit machines in a different order from one another. We would be looking for an efficient method of scheduling the job-shop in hopes that it could be easily extended to the more general FMS variant. The contribution deterministic scheduling has made so far is to assure us that efficient optimal seeking methods for job-shops very likely do not exist. They belong to a class of "difficult" problems whose time to solution is not bounded by a polynomial function of the parameters (number of machines and parts). This class is said to be NP-Complete [Coffman, 1976]. Because FMS's seem to be magnitudes more complex than job-shops, and because the number of parts to schedule can be quite high, pursuing an optimal solution seems ill-advised and to date no one has attempted it.

Hueristic methods developed from job-shop research are feasible, though, and some work has been done to apply them to FMS's. Stecke (1977) explores the utility of decision rules for scheduling parts through the system. In her work, a very simple decision structure based on machine priority and part priority is set up. Part priorities are assigned by standard dispatching rules (Shortest Processing Time (SPT), Longest Processing Time (LPT), etc.) and machine priorities are assigned in accordance with their total work load. If a number of machines are idle, parts waiting for the machine with the highest priority are ranked to determine which would be processed next. That part with highest priority is chosen. In all, sixteen different dispatching rules were tried with the result that SPT/TOT (Shortest Processing Time for current operation divided by the total processing time of the part) seemed to perform best relative to the other rules.

Schedule generation based on a priority structure is often used and has a number of clear advantages. They are easy to implement, consume minimal computer time and are very adaptable to unforeseen disturbances. But there is never a clear connection between the structure and the global concerns of the system. Finding good schedules within this structure forces the analyst to use the "shotgun" method for problem solving. That is, try every conceivable rule and choose the one that generates the best schedule. We suggest in following chapters that some decisions in the FMS problem can be made effectively using priority decision rules and some cannot.

The users of FMS's (there might be a total of twenty FMS's in operation over the whole world) schedule their systems in some way but their work is not well documented. One very good treatment (Hutchinson,1977) confirms our belief that quick, simple rules incorporating the peculiarities of each system are generally employed.

Queueing Theory

The first direct application of queueing theory to FMS's is due to Solberg (1978). Using the theory for closed networks as developed by Jackson (1963) and Gordon and Newell (1967), he constructs an analytical model for system behavior that shows good agreement with those performance measures obtained from simulating actual FMS's. Although the model must make several unrealistic assumptions (equilibrium behavior, exponential service times, etc.), its accuracy does not seem to be overly sensitive to minor violations of these assumptions. Some theoretical justification for this will appear in a forthcoming publication (Suri,1980).

Another work that uses this same network of queues theory as part of an optimal work allocation scheme is performed by Secco-Suardo (1978). A formulation is given for finding the optimal routing (i.e., our definition of routing) for each setup. However, only one class of job is allowed within the network. In order to account for pixture constraints on each setup, an important practical concern, an approach that associates each setup type with a separate class must be taken. Baskett et. al. (1975) have extended the theory to multi-class networks,but numerical difficulties can hamper its implementation (Reiser and Lavenberg,1978). So treating a multi-class problem using the approach in (Secco-Suardo,1978) raises serious questions about the feasibility of arriving at an optimal solution for larger FMS's.

An approach that uses elementary queueing theory within a network flow optimization scheme is discussed by Kimemia and Gershwin (1979). There seem to be no numerical difficulties associated with this method but here again constraints on pixtures cannot be specified, and machine failure is not considered.

1.5 Outline for Research

The FMS scheduling problem assumes an awkward position relative to analytical techniques that are available. Queueing theory is global in outlook and can be used for overall resource work balance, but it does not have the resolution to see the effects of different sequences of parts. On the other hand, deterministic scheduling theory can resolve the difference between different sequences, but the capacity of the available computer does not allow for a global treatment of the problem.

This thesis attempts to combine the two approaches in a framework for decision making such that they complement one another. In addition, we extend the state of the art in modeling FMS's to include machine failure and limitations on pixtures.

The second chapter begins by examining decision variables for the scheduling problem and how they can be organized to reflect a global objective. A multi-level structure for decision making emerges from a decision partitioning concept. Each level of the structure solves for certain decisions of the problem until, finally, all decisions are fully resolved. We propose a three level framework for the FMS considered in this thesis.

The algorithm for the first level in the structure is developed in Chapter 3. A recent advance in networks of queues theory (Mean-Value Analysis) with very efficient solution procedures is used to model an FMS as a multi-class network. This model is embedded in an optimization scheme that minimizes the completion time of a given production target assuming machines fail according to some probablistic model. The algorithm provides a fixed schedule that specifies the optimal mix of setups to input to the system as a function of which machines have failed. Also, the optimal setup routing is determined.

Chapter 4 discusses the Second and Third Level. After the mix of setups to be input has been determined, Level 2 undertakes the task of sequencing those setups into the system. A dynamic program that uses a minimum of information is constructed to do this. Loosely speaking, the objective of this algorithm is to deliver to each resource a steady supply of work as dictated by Level 1. For this reason, it has been dubbed the "Even-Flow Algorithm."

Eventually, every task must be assigned to a particular resource at a particular time. Level 3 accepts this responsibility. It incorporates decision rules in a real-time environment.

The performance obtained from the multi-level decision structure and its component algorithms is studied in Chapter 5 for FMS's of reasonable size. The effects certain unmodelled transient phenomenon have on performance are discussed and experimentally observed.

CHAPTER 2

STRUCTURING THE FMS SCHEDULING PROBLEM

2.1 Introduction

Having introduced the FMS scheduling problem in the previous chapter, we must now construct a formulation to solve it. Considering the dimensionality of the problem, research has shown that the time needed to solve an exact formulation is prohibitively long. Suboptimal solution procedures must be constructed that yield "good" results while remaining within the computational constraint imposed.

This chapter outlines a multi-level approach for solving the FMS scheduling problem where each level is responsible for resolving certain decisions. Through a succession of levels, eventually all decisions are made and the problem is solved, albeit a suboptimal solution. The computational requirements for this structure are lower than that of an exact formulation because all decisions are not solved in a fully coordinated way. After partitioning the decisions of the problem into groups, each group is treated in turn. The performance of the multi-level decision structure depends entirely on how decisions are defined and partioned.

Preliminary to defining the decisions of the problem and placing them in a structure, it is appropriate to introduce the framework by which we view machine failure. It influences the way in which decisions are defined. A fundamental happening we call a system failure condition is examined next.

2.2 Machine Failure Dynamics

When a machine breaks or is repaired, we say that the system enters a new failure condition. A failure condition is characterized by those machines that are currently down and, as time goes on, the system shifts from condition to condition. A Condition Transition Diagram for a three-machine problem is shown in Figure 2.1.

Since machine failure is a random process, predictions of future behavior can be described in probabalistic terms only. Assuming some initial starting point, there is an associated probability of finding the system in each failure

Chapter 3 to help make rational work tradeoffs among the various failure conditions. It is expected the best estimates will be obtained from a combination of historical data, preventive maintenance schedules as well as any analytical models that may be devised.

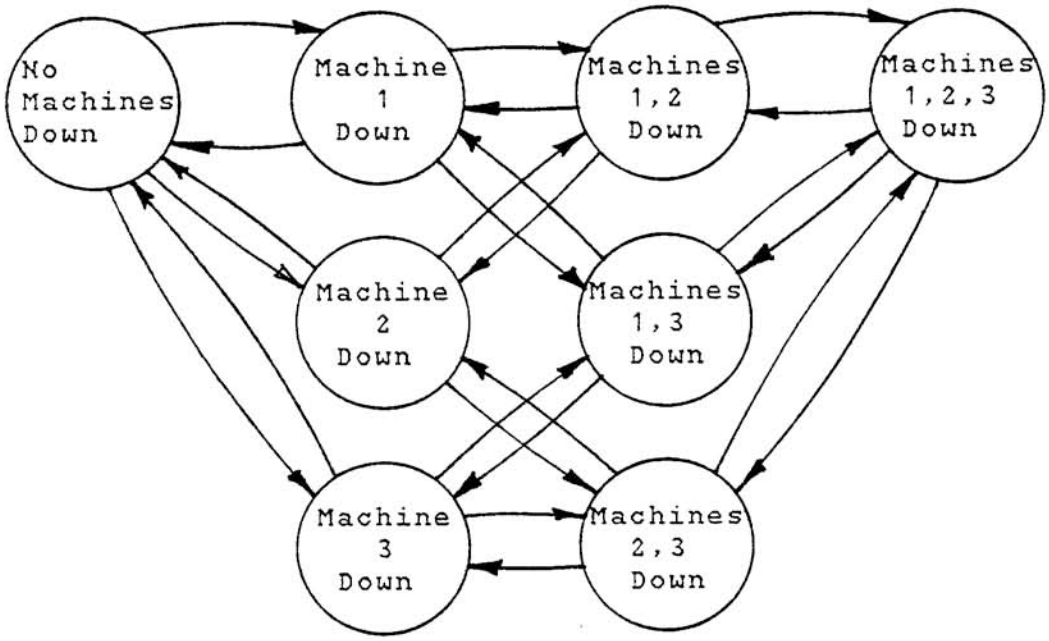


Figure 2.1: Failure Condition Transition Diagram

2.3 Multi-Level Solution Structure

One general representation for the decisions of our problem has already been given and appears in Section 1.3.3 as a set of "control variables". The four classes of control are repeated here for reference:

1. Sequence of setups into the system.
2. Loading Time for each setup entering the system.
3. Next Operation for each setup.
4. Next Machine for each setup.

The representation is complete in the sense that no other decisions need be made in order to fully resolve the problem. However, other representations are possible as well. Figure 2.2 shows an alternate set of decisions we shall use within a three-level decision-making structure.

Aside from the explicit dependence of decisions on failure condition, the new representation differs from the old in that the sequencing decision is carried out in two steps. First, the mix of setups is determined for each failure condition, then the sequence for setups comprising this mix is found. All other decisions correspond to the original representation.

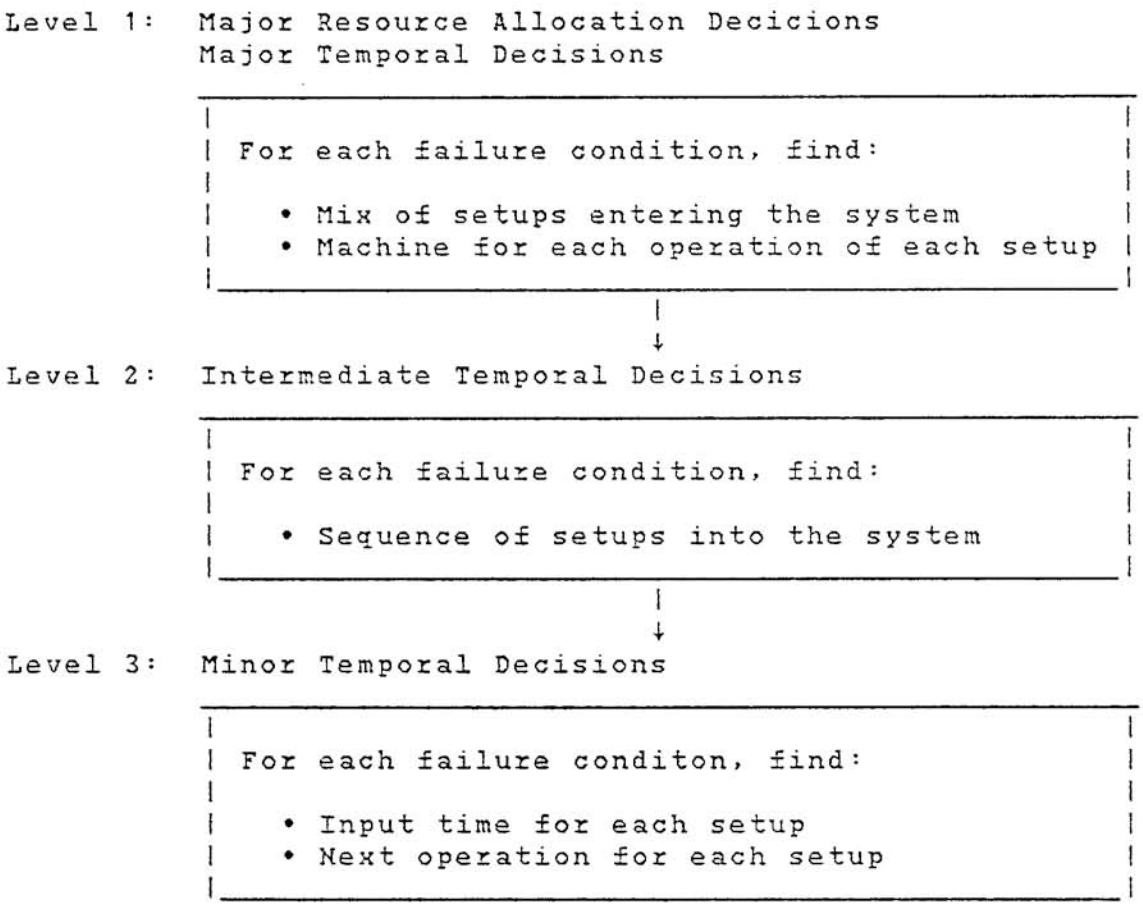


Figure 2.2: Decision Structure for FMS Problem

The structure indicates how the problem is simplified. The top level makes those decisions that have a major impact on the overall system objective while the decisions of the last level have a minor impact. In general, the way in which decisions should be partitioned is not easily quantified and thus relies heavily on the experience of the analyst or trial and error. Dynamic resource allocation problems, however, suggest two important quidlines for structuring decisions. They are founded on a classification of decisions into two types: resource decisions and temporal decisions.

A resource-decision is concerned with a choice among resources (e.g., the allocation of a task to one of several machines) while a temporal-decision is concerned with the time a particular task is allocated to a resource (e.g., determining the order in which the operations of a setup are to be performed). The decisions of a dynamic resource allocation problem can generally be placed in one of these two classes. In our problem, the "machine allocation" decision is the only resource decision, all others are temporal decisions.

1. The Relative Importance of Resource Decisions

The system throughput is determined mainly by the resource(s) with the highest workload. Since our system objective is directly related to throughput, the resource-decisions associated with these bottlenecking (or pacing) resources have high impact and should be treated first, before any others. Decisions associated with resources of high capacity are less important to the problem and can be put off to a lower level.

Our decision structure reflects an FMS whose machines have a workload at least as high as that for the material handling system. Thus, the decisions associated with machine allocation are treated at the first level. If the capacity of the machines far exceed that of the MHS, then the machine allocation decision is of little consequence and can be made after all others with little impact on the overall objective. The relative importance of resource allocation decisions, then, is easily obtained by ranking the workload for each resource.

2. Ordering Temporal Decisions

Temporal decisions should be defined in a manner that makes determining an ordering very natural. In our problem, the mix of setups entering the system is found

first, then the sequence for those setups is determined, and finally, very detailed information about when setups are loaded and the order in which their operations are performed is specified.

We are steadily narrowing the time at which a task is to be allocated to its resources. First, each setup is allocated to broad time periods characterized by a particular failure condition. Then those setups associated within each failure condition are sequenced. This further narrows the time at which they are to occupy resources. Finally, each operation of each setup is allocated to its assigned resource at a particular time. This notion of continuously narrowing the alloction time for setups was the basis for expanding the original sequencing decision as we did.

2.4 Performance Measure for Each Level

There is, associated with each level in the structure, a mathematical formulation for resolving decisions. The rationale for making decisions comes from the performance measure incorporated within each formulation. Ideally, each level chooses as its objective the same overall system objective. In our case, this would be to minimize the completion time of the production target. In practice, this is often not convenient or even possible. The manner in which decisions are defined, or the need to further simplify the solution procedure can force the adoption of alternate objectives. When this is the case, an effort should be made to relate the alternate objective to the original system objective.

The performance measures we shall use in the decision structure are:

Level 1 - Minimize the completion time of the production target.

Level 2 - Maximize average production rate during each failure condition.

Level 3 - Minimize the delay encountered by each setup as it proceeds through the system.

The exact meaning of each of these objectives and the reason for their form will be made more clear when developed in the context of the actual mathematical formulation.

2.5 A Profile of Each Level

We shall preview what is to come in succeeding chapters by giving a profile of the formulation at each level. These profiles will bring out the main features and indicate, in broad terms, how information flows between levels. Profiles are comprised of the following five characteristics:

1. An objective function.
2. Mathematical techniques used in formulating the problem.
3. Inputs required for the mathematical formulation.
4. Problem data needed for the formulation.

   The distiction between "Inputs" and "Problem Data" is somewhat artificial. However, it is usefull in the organization of the problem.
5. Outputs from the formulation.

Level 1

Objective	Minimize completion time for production target.
Techniques	Nonlinear mathematical programming.
	Networks of queues theory.
Input	Production target.
Problem Data	Maximum number of each pixture type.
	Maximum number of setups allowed in the system.
	For each operation, all machine alternatives and associated processing times.
Output	Setup mix for each failure condition.
	Setup routing for each failure condition.

Level 2

Objective	Maximize average production rate during each failure condition.
Technique	Dynamic programming.
Input	Setup mix for each failure condition. Setup routing for each failure condition.
Problem Data	Not applicable. Aside from the input, no other information is needed.
Output	Sequence of setups into the system for each failure condition.

Level 3

Objective	Minimize delay for each setup as it travels through the system.
Technique	Dynamic decision rules.
Input	Sequence of setups into the system.
Problem Data	Detailed information about the current state of the system. The exact information required depends upon the decisions rules used. Examples include setup processing times, number of parts enqueued before each machine, travel times, etc.
Output	Loading time for the next setup. Next operation for each setup after having had an operation performed.

Notice that the output of each level is considered as input to the following level.

2.6 Information Feedback

The previous section highlights the two types of information needed to determine the output of each level: input information and problem data. As time goes on, unexpected changes in either type may justify re-solving the level affected. Since the output of one level serves as input to another, a chain reaction can occur where all succeeding levels musts be re-solved.

Aside from simply updating problem data, we shall periodically re-solve the Level 1 formulation on the grounds that its knowledge of future machine failure is not perfect. As we shall see in Chapter 3, a scheduling strategy is devised based on expected machine failure behavior. If actual machine failure does not conform to our expectations, then the computed setup mix and routing will, in fact, not be optimal. Rather than continue controlling the FMS with an incorrect scheduling strategy, it is better to stop and determine another based on those setups yet to be processed from the original production target. It may be necessary to obtain new schedules several times in the course of a production run. This process of obtaining new strategies with updated information is called optimal open-loop feedback (Dreyfus and Law,1977) We shall discuss this in greater detail in Chapter 3.

CHAPTER 3

LEVEL 1: SETUP MIX AND SETUP ROUTING

3.1 Introduction

The model and algorithm resident at Level 1 are developed in this chapter. Before the details are revealed, it is helpful to motivate some of the development from a physical point of view. Scheduling FMS's is the general concern of this research, but scheduling them, when machines are allowed to fail, is of particular interest. So we ask the question: What should be done when a machine fails?

The dynamics of machine failure carry the system from one configuration to another. It seems natural to partition the time horizon for the problem into a series of disjointed segments. Each segment corresponds to a period during which the configuration of the system remains constant. We can then make informed scheduling tradeoffs among those individual time intervals by predicting future machine failure with a probabalistic model. As we shall see, an interesting effect surfaces. A scheduling strategy is devised that can be interpreted as anticipating future machine failure. Depending on the failed machine, certain parts may be deferred until some later time when conditions are perceived to be more favorable. The reason for this is that, from a processing point of view, parts see the system differently depending upon which machines are down.

We begin with the Level 1 formulation that predetermines setup mix and setup routing for each failure condition. It is a nonlinear mathematical program with an embedded network queueing model. Following this, appropriate algorithms are developed to solve the formulation.

3.2 Level 1 Formulation

One reminder before a number of definitions are stated: a setup can proceed through an FMS on one or possibly more routes. A route is characterized by those machines visited by the setup as it passes through the system.

Define:

I = Set of system failure conditions that are to be considered in the problem formulation.

S = Set of distinct setup types to be processed.

R_{si} = Set of routes setup s can take through the system during failure condition i. $i \in I$, $s \in S$

N_s = Total number of s setups to be processed.

$\theta(t)_i$ = Actual time the system spends in failure condition i over a time period of length t.

$P(t)_i$ = Expected proportion of time the system spends in failure condition i over a period of length t:

$$P_i(t) = E\{\theta(t)_i\}/t$$

The expectation is performed over an ensemble of "outcomes" for vect(θ).

λ_{sri} = Rate of production of setup s using route r during condition i.

T = Time to complete processing of all setups.

Our objective is to minimize the time it takes to process all setups. This completion time is a function of machine failure, the production target, and other parameters whose influence is felt through vect(λ). Our problem can be written in the general form:

$$\underset{\vec{\lambda} \in L}{\text{MINIMIZE}} \quad T(\vec{\theta}(T), \vec{N}, \vec{\lambda}) \tag{3.1}$$

where L is a complicated constraint set that depends on the number of pixtures available for each setup. Note that vect(θ) actually depends on the completion time itself.

The problem is not well posed, though, because there is no way to predict vect(θ) with certainty. Our only recourse is to minimize the expected completion time:

$$\underset{\vec{\lambda} \in L}{\text{MINIMIZE}} \quad E\{T(\vec{\theta}(T), \vec{N}, \vec{\lambda})\} \tag{3.2}$$

But this too is not promising because the functional form for T is not apparent, so formally taking an expectation is out of the question.

In order to continue, we must make the assumption that T is approximately linear within the range of variation of vect(θ(T)), the only stochastic component of the problem. The expectation operator can then be brought within T and applied to vect(θ(T)) directly. The result is:

$$\underset{\vec{\lambda} \in L}{\text{MINIMIZE}} \quad T(E\{\vec{\theta}(T)\}, \vec{N}, \vec{\lambda}) \tag{3.3}$$

or

$$\underset{\vec{\lambda} \in L}{\text{MINIMIZE}} \quad T(P(T)T, \vec{N}, \vec{\lambda}) \tag{3.4}$$

A representation that brings out more clearly the dependency on problem parameters is writen as follows:

$$(P1) \qquad \text{MINIMIZE } T$$

$$\vec{\lambda}, T$$

s.t.

$$\sum_{i \in I} \sum_{r \in R_{si}} \lambda_{sri} P_i(T) \; T \geq N_s \qquad s \in S,\ i \in I,\ r \in R_{si} \tag{3.5}$$

$$\vec{\lambda} \in L \tag{3.6}$$

This formulation attempts to choose production rates for each route of each setup during each system failure condition such that the time to process all setups is minimized. The first set of constraints insure that the required number of each setup are produced over the time horizon for the problem.

An alternate form for P1 that addresses the problem of defining L is developed by introducing new decision variables as follows.

Define:

n_{sri} = Average number of pixtures devoted to route r of setup s during condition i.

$\vec{n}_i = \{n_{11i}, n_{12i}, \ldots\}$

$\vec{\tau}(n)_{srm}$ = Average total time setup s, using route r, spends at machine m. This includes service time and queueing time. Note that τ is a function of the total pixture allocation for a particular failure condition. This relationship will be defined in Section 3.4.

PMAX = Maximum number of pixtures to be resident within system during any failure condition.

$FMAX_s$ = Maximum number of pixtures available for setup s.

Using Little's result for relating the number of setups resident in the sytem to flow rate (Kleinrock,1975):

$$\lambda_{sri} = n_{sri} / \Sigma_m \tau_{srm}(\vec{n}_i), \qquad (3.7)$$

P1 is rewritten as follows:

(P2) $\underset{\vec{n},T}{\text{MINIMIZE}}\ T$

s.t.

$$\frac{N_s}{\sum_{i\in I} P_i(T) \sum_{r\in R} \frac{n_{sri}}{\sum_m \tau_{srm}(\vec{n}_i)}} \le T \quad \text{each } s \qquad (3.8)$$

$$\sum_r n_{sri} \le FMAX_s \quad \text{each } s,i \qquad (3.9)$$

$$\sum_s \sum_r n_{sri} \le PMAX_i \quad \text{each } i \qquad (3.10)$$

$$n_{sri} \ge 0,\ T \ge 0$$

Problem P2 essentially balances the workload for all setups over their routes and over the failure conditions considered. One unit of each component of vect(n) carries a demand for certain resources in some amount. The object is to choose vect(n) such that the production target is achieved (constraints 3.8), while minimizing the completion time for the production target. Note that Little's equation assumes average values for the components of vect(n), so they need not be constrained to be integers. Constraints 3.9 and 3.10 specify limits on the number of each setup available and the total number allowed in the system, respectively.

Aside from the nonlinearity of constraints 3.8, the only other difficulty that could hamper the solution of P2 lies in the evaluation of the terms τ sub(srm) contained in those constraints. These machine visit times are, themselves, nonlinear functions of the allocation, vect(n). Unfortunately, analytic expressions for these nonlinear functions are not available. Problem P2 could be solved in a straightforward manner using standard methods if they were available. Section 3.4 discusses an iteritive technique derived from networks of queues theory that we shall use to evaluate vect(τ). The technique is known as Mean Value Analysis (MVA).

An algorithm for solving P2 will be outlined subsequently. Before doing so, a formulation alternate to P2 is introduced and MVA is discussed. The alternate formulation is valid when the number of available pixtures is not constrained and the system is allowed to saturate. (The system is saturated when the bottleneck resource is busy 100% of the time.) It is very easy to solve and will be used to simplify the solution of P2.

3.3 An Approximate Formulation for Level 1

Our nonlinear formulation must be employed when the number of each pixture type is constrained or limits are placed on the total number of pixtures allowed in the system. If these constraints are relaxed, it is then possible to construct a formulation that is almost linear and easier to solve than problem P2. This interests us because some systems may indeed operate with little constraint on pixture availability. Beyond this, the simplified formulation can be used to provide a reasonable starting point for the algorithm for the more interesting nonlinear problem.

Define the following:

x_{sri} = Number of setups using route r during condition i.

t_{srm} = Process time for route r of setup s on resource m.

M'_i = Set of resources that are operational during state i.

As before, we wish to minimize the completion time of the production target. When pixture constraints are relaxed, this is accomplished with the following program:

(P3) $$\underset{\rightarrow}{\text{MINIMIZE } T}$$
$$x, T$$

$$\text{s.t.} \quad \sum_i \sum_r x_{sri} \geq N_s \qquad \text{each } s \qquad (3.11)$$

$$\sum_s \sum_r x_{sri} t_{srm} \leq P_i(T)T \qquad \text{each } i,\ m \in M'_i \qquad (3.12)$$

$$x_{sri} \geq 0 \qquad \text{each } s,r,i \qquad (3.13)$$

The first set of constraints insure that the production target is met while the second set specifies that the work allocated to each failure condition is less than the actual expected time available during that condition. Strictly speaking, x sub(sri) should be constrained to the integers. However, these values are generally quite high, the error involved in rounding the linear solution to the nearest feasible, integer solution is small and will be ignored.

The solution of P3 is complicated somewhat by the nonlinearity of T in constraints 3.12. It can be solved quite easily, though, through an iterative procedure where a series of linear programs are solved. The details of the solution procedure have been relegated to Appendix A in order that we may proceed with the development and algorithm for P2 without distraction.

3.4 Mean-Value Analysis

3.4.1 Introduction

Various methods exist for obtaining the machine visit times (τ) for a given allocation of pixtures vect(n). Brute-force simulation of the desired system is one such method that is highly accurate but very costly in terms of the computation power required. We shall use a technique derived from Networks of Queues Theory that yields approximate results for systems with determinisitic process times but is very simple relative to simulation. Where simulation may require 15 seconds of computer time for one evaluation of vect(τ), MVA may require less than 0.1 second. We shall see that the loss in accuracy in going to MVA is very small for a wide class of systems.

MVA is a rather recent development in networks of queues theory that was first presented by Reiser and Lavenberg (1978), and extended by Bard (1978). The method considers closed networks, where a fixed number of jobs circulate among the nodes (machines), and yields other performance measures as well (e.g.,throughput, utilization, average queue size, etc.). It takes its name from the fact that it deals mainly with the first moment (mean value) of distributions associated with the underlying probability space of the problem. The analysis is based on a relation between the mean waiting time at a machine and the mean queue lengths for a system with one less job. This "mean-value" equation, together with Little's result applied to each setup and separately to each machine, furnish the needed equations on which to base a suitable algorithm.

One advantage MVA has over other queueing techniques is its computational simplicity and easy implementation for multi-class networks. In multi-class networks, jobs are associated with one of several classes depending on their distinguishing characteristics. By classifying jobs according to setup, route and failure conditions (i.e., the jobs associated with each n sub(sri) constitute individual classes), MVA enables us to predict system behavior on a class-by-class basis. This is precisely what P2 requires. Another advantage is that its associated algorithm has a physically meaningful interpretation that is helpful for extending the analysis to systems that are somewhat closer to actual FMS networks.

A drawback to MVA that is common to most queueing techniques are the assumptions it requires to yield exact results. They are:

Assumptions

1. Under the condition that the queueing discipline is FCFS (First Come First Served), the service times must be exponentially distributed and independent of a setup's class membership.

2. The system is assumed to have reached probabalistic equilibrium.

   Probablistic equilibrium for a network of queues is the condition reached when the probability of finding a given number of setups in all queues is a constant fuction of time. This is the standard definition adopted by queueing theorists.

The FMS's we consider have FCFS queueing disciplines so the first assumption is applicable. However, the properties of most FMS's lie in strict violation of the first assumption. Since the service times for setups are fixed at constant values and since a particular setup on a particular route during a particular failure condition constiutes a class, the stochasitc factor needed to satisfy queueing theory is not present. Regardless, we shall forge ahead and demonstrate that quite reasonable predictions of performance can be obtained using the theory in a heuristic fashion.

The second assumption is potentially painful. Incorporating MVA into P2 implies that the system moves instantaneously from one equilibrium condition to another as the failure condition changes. For any dynamic system, a transition period will always follow an abrupt change in the configuration or control of the system. If these periods are long enough and the occurance of disturbances often enough, equilibrium may never be established. Chapter 5 examines this issue in more detail and offers some evidence that suggests equilibrium assumptions, although rarely satisfied, yield adequate performance in many cases.

3.4.2 MVA Algorithm

A complete development for the MVA algorithm is given in Appendix B. This section merely presents the final result and demonstrates its accuracy by a small example.

Definitions:

n_s : Population size of setup s. (Eventually, a class will be associated with each component of vect(n). During the MVA development, routes and failure conditions are not distinguished.)

$\vec{n} = (n_1, n_2, \ldots, ns)$: Population vector.

$t_{s,m}$: Service time of setup s at machine m.

q_m : Equilibrium mean queue size at machine m.

$q_{s,m}$: Equilibrium mean number of setup s at machine m.

$\tau_{s,m}$: Equilibrium mean waiting time (including service time) of setup s at service center m.

λ_s : Setup s throughput.

e_i : |S|-dimensional unit vector in the ith direction.

M_s : Set of machines setup s visits for processing.

Also, vect(n) may appear as an argument for certain quantities, indicating that they are evaluated for a network with population vect(n), e.g.:

$$q_m(\vec{n})$$

for mean queue size.

Algorithm MVA

1. Initialization: $q_{s,m} = n_s / |M_s|$ for all m in M_s,
   $q_{s,m} = 0$ for all m not in M_s.

2. Repeat steps (3) to (6) until a suitable convergence criterion is met.

3. $$\epsilon_{s,m}(j) = \frac{q_{j,m}(\vec{n})}{\sum_i q_{j,i}(\vec{n})} \quad s = j$$

   0 otherwise.

4. $\tau_{s,m} = t_{s,m} + \sum_j t_{j,m} (q_{j,m} - \epsilon_{j,m}(s))$ Mean Value Eqn.

5. $\lambda_s = n_s / \sum_m \tau_{s,m}$ Littles Result for Setups

6. $q_{s,m} = \lambda_s \tau_{s,m}$ Littes Result for Machines

The operation count per iteration cycle for this algorithm can be manipulated to 9*|S|*|M| + |S|. Experience has shown that the number of iterations required such that the change from one iteration to the next is less than 1% is on the order S/2 when the procedure is started from an arbitrary point. When given a good start, far fewer iterations are needed. This will be significant in the algorithms to follow. The complexity of the algorithm then, is $O(S^2)$.

In addition to its simplicity, another virtue of Algorithm MVA is that the population of each class is not constrained to be integer, or even greater than one for that matter. This enables us to specify an average population size for each class that is fractional. Since the average population of individual classes for an optimal solution is not expected to be integer, this feature is most welcome.

3.4.3 MVA Test Results

The MVA algorithm was run and compared against results obtained from simulation. The system simulated had 4 machines, a limit of six resident pixtures, a conveyer MHS, and processed an unlimited number of 10 different setups. The operation times for each setup were chosen before the run from a uniform distribution with range 1.0 minute to 29.0 minutes and fixed throughout the simulation. These operation times were used by algorithm MVA for t sub(sm).

The difference between MVA and simulation results were recorded for the main performance measures and averaged over 50 runs. (Each run differed only in the values of process times for each setup.) They appear in Figure 3.1.

The results show excellent prediction for average queue size but throughput and utilization are biased somewhat below that found for the simulation. This pessimistic prediction is probably rooted in the exponential distribution MVA assumes for operation service times. Very large values that an exponential distribution will yield for operation times, albeit with low probability, are not experienced by the simulation. Any time there is a wide variation in service times, the internal disruption caused for setup flow will lower throughput and utilization. However, the relative utilizations (obtained by scaling the highest utilization up to 1.0 for both the MVA prediction and simulation result) are predicted accurately as shown in Figure 3.2. Considering that the bias is generally quite low (within 11%), it is expected that the optimal work-balancing decisions will not be thrown off by using MVA as a model for the system.

PERFORMANCE MEASURE	% DIFFERENCE 100*(SIM-MVA)/SIM		TYPICAL RUN	
	MEAN	STD.DEV.	SIM	MVA
Throughput (parts/minute)	11.0	5.8	.0526	.0427
Utilization				
Machine 1	10.7	6.1	75.0	72.0
Machine 2	9.6	6.9	63.0	59.0
Machine 3	8.9	7.4	79.0	70.0
Machine 4	10.0	7.1	51.0	46.0
Average Queue Size				
Machine 1	0.4	15.0	1.4	1.7
Machine 2	-3.4	13.9	0.9	0.9
Machine 3	0.3	14.4	1.4	1.3
Machine 4	2.7	11.0	0.8	0.6

Figure 3.1 MVA Test Results

	% DIFFERENCE	
	MEAN	STD DEV.
Machine 1	-1.4	6.7
Machine 2	-0.6	5.5
Machine 3	-1.5	5.1
Machine 4	-2.4	8.7

Figure 3.2 Relative Utilizations from MVA

3.5 Algorithm for Problem P2

Many nonlinear programming algorithms exist which could be brought to bear on P2, even though the evaluation of vect(τ) may be somewhat involved. The simple form of the objective and the few nonlinear constraints of the problem, though, suggest an algorithm that appears in (Suri,1978). The algorithm is a variation on the successive linearization method and is briefly described now.

3.5.1 Modified Successive Linearization

The Level1 problem has the following form:

$$\text{MINIMIZE}_{\vec{n},T} \; T$$

$$g_s(\vec{n},T) \le T \qquad \text{each } s \qquad (3.14)$$

$$\sum_r n_{sri} \le FMAX_s \qquad \text{each } s,i \qquad (3.15)$$

$$\sum_s \sum_r n_{sri} \le PMAX \qquad \text{each } i \qquad (3.16)$$

$$n_{sri} \ge 0 \qquad T \ge 0$$

Starting from some feasible solution, we intend to iteratively reduce T by some small amount and find the change in pixture allocation required to satisfy all constraints. When it is no longer possible to reduce T and obtain a feasible allocation, we have reached a solution and the process is stopped. The change in T at each step is small enough that a linearized version of the original problem can be solved for the change in allocation. Thus, the original nonlinear problem is reduced to a series of linear ones.

Initial Starting Point

The algorithm must start from some feasible point $\vec{n}^0$ and T^0. A good starting point for vect(n) can be easily derived

by chosing an allocation that is both feasible and corresponds to the routes that were singled out in the approximate formulation (problem P3). Having chosen vect(n^0), a feasible T is always available merely by setting it to some large value. However, the higher T^0 is, the more iterations the algorithm must go through to reach optimality.

The following successive substitution method for finding T^0 may be tried although it is not guaranteed to converge:

$$T^0_{k+1} = \max_s g(\vec{n}^0, T^0_k) \qquad T^0_0 \text{ arbitrary} \qquad (3.17)$$

However, even if it does not converge, the above equation can be used to help bound the solution. The following Lemma states a fudamental property of equation 3.17.

Lemma 3.1

Let T* be a solution to equation 3.17.

$$\text{If } T^0_{k+1} < T^0_k \quad \text{Then } T^* \leq T^0_k$$

$$\text{If } T^0_{k+1} > T^0_k \quad \text{Then } T^* \geq T^0_k$$

Proof: Place g sub(s) in a form corresponding to equation 3.8 in P2 and write equation 3.17 as:

$$N_{s'} = T^0_{k+1} \sum_i P(T^0_k)_i \, c_{is'} \qquad (3.18)$$

$$s' = \arg\max_s g$$

where $c_{is'}$ are constants that depend on $\vec{n}^0$ and $N_{s'}$.
(They are nondimensional setup flow rates.)

If $T_{k+1} < T_k$, then

$$N_{s'} \leq T_k \sum_i P_i(T_k)\, c_{is'} \qquad (3.19)$$

The time spent in failure condition i during a time period of length T is:

$$T\, P_i(T) \qquad (3.20)$$

From physical considerations, this occupation time is a nondecreasing function of T. Therefore, equation 3.19 immediately yields:

$$T^* \leq T_k$$

If $T_{k+1} > T_k$, then

$$N_{s'} \geq T_k \sum_i P_i(T_k)\, c_{is'} \qquad (3.21)$$

and

$$T^* \geq T_k$$

***

The algorithm for problem P2 proceeds as follows:

Algorithm P2

Step 1: Set $k = 0$; choose an $\vec{n}^0$ and find T^0, using successive substitution as indicated by Equation (3.17).

Step 2: (for $k > 0$ only)

$$\text{Set } y^{k'} = \underset{s}{\text{MAX}}\, g_s^k(\vec{n}^k, T^k) \tag{3.22}$$

$$\text{If } y^k < T^{k-1}, \text{ then let } T^k = y^k \text{ and continue.}$$

Otherwise, stop. Solution is:

$$\vec{n}^{k-1}, T^{k-1}$$

Step 3: Evaluate $A_s^k(\vec{n}^k, T^k)$

where:
$$A_s^k = \frac{\partial g_s(\vec{n},T)}{\partial \vec{n}} \Big|_{\vec{n}=\vec{n}^k;\, T=T^k} \tag{3.23}$$

Step 4: Choose ΔT^k and set:

$$T^{k+1} = T^k - \Delta T^k \tag{3.24}$$

$$\text{Evaluate } g_s(\vec{n}^k, T^{k+1}) = g_s^{k+1} \tag{3.25}$$

Step 5: Determine $\Delta \vec{n}^{k}$ from the following linearization of P2.

$$\Delta \vec{n}^{k} = \arg\min_{sri} |\Delta n|$$

s.t.

$$g_s^{k+1} + A_s \Delta \vec{n}^{k} \leq T^{k+1} \quad \text{each } s \qquad (3.26)$$

$$\sum_r (n_{sri}^{k} + \Delta n_{sri}) \leq FMAX_s \quad \text{each } s,i \qquad (3.27)$$

$$\sum_s \sum_r (n_{sri}^{k} + \Delta n_{sri}) \leq PMAX \quad \text{each } i \qquad (3.28)$$

$$n_{sri}^{k} + \Delta n_{sri} \geq 0 \quad \text{each } s,r,i \qquad (3.29)$$

Step 6: Set $\vec{n}^{k+1} = \vec{n}^{k} + \Delta \vec{n}^{k}$, (3.30)

Set $k = k+1$

Go to Step 2

Choosing ΔT for fast convergence is an exercise in magic. Our technique initially sets ΔT to some large value (e.g., $\Delta T = T/4$). As long as substantial improvement in the objective is realized, (e.g., $T(k+1) - T(k) \leq \Delta T(k)*3/4$), ΔT is not changed. If the goal for improvement is not met, ΔT is reduced (e.g., $\Delta T(new) = \Delta T(old)/2$), never to be increased again. The increment is not reduced below some fraction of T (e.g., $\Delta T \geq 0.01*T$) in order to keep solution time within reasonable bounds. Choosing ΔT means choosing the various multiplicative factors parenthetically introduced above.

3.5.2 Computation of Gradients

The above algorithm requires the gradients of the nonlinear constraints with respect to pixture allocation:

$$\frac{\partial g_s(\vec{n})}{\partial \vec{n}} \tag{3.31}$$

After chaining through several intermediate variables, eventually the gradients of the queue lengths for each setup on each of its routes on each machine is required:

$$\frac{\partial q_{srm}}{\partial \vec{n}} \tag{3.32}$$

However, only the implicit relation (Equation B.15, Appendix B) derived from Algorithm MVA exists for determining the individual queue lengths, so their gradients are not easily obtained. If a vector is constructed from the q's as follows:

$$\vec{q} = (q_{111}, q_{112}, \ldots) \tag{3.33}$$

then a vector version of Equation B.15 may be written as:

$$\vec{q} = \vec{f}(\vec{q},\vec{n}) \tag{3.34}$$

The desired gradients are obtained by expanding this equation in a Taylor series about a given vect(n), then dividing by an increment vect(Δn) and taking the limit as that increment approaches zero. The result is:

$$\frac{\partial \vec{q}}{\partial \vec{n}} = \left[I - \frac{\partial \vec{h}}{\partial \vec{q}} \right]^{-1} \frac{\partial \vec{h}}{\partial \vec{n}} \tag{3.35}$$

The difficulty of this method lies in the size of the matrix that must be inverted, presumming the matrix is not singular. For typical problems, it may have rank 100 or more.

The only alternative is to proceed in standard brute-force manner. Increment each independent variable in turn from the current point and form an approximate partial derivative from the differences obtained. This is shown symbolically here:

$$\frac{\partial g_s(\vec{n})}{\partial n_i} = \frac{g_s(\vec{n} + \Delta n_i \vec{e}_i) - g_s(\vec{n})}{\Delta n_i} \qquad (3.36)$$

where vect(e(i)) is, the unit vector in the ith direction.

One application of Algorithm MVA is necessary to obtain the partial derivatives for all components of vect(g) with respect to one component of vect(n). Since the dimension of vect(n) is:

$$\sum_s \sum_i R_{si}$$

the computational burden could become high for large problems. There are three methods for reducing this labor, route elimination and failure condition elimination to be discussed in the next section, and approximate gradients.

Approximate gradients are easily obtained from Algorithm MVA by relaxing the convergence criteria so that fewer iterations are needed. This is appropriate in the early cycles of Algorithm 3 when it is presumed the cost function is steep and precise evaluations of g(n) are not needed to obtain a general direction for improvement. However, as the optimal point is approached, the components of the gradients in the feasible direction become smaller and much more sensitive to errors in their evaluation. A labor saving strategy relaxes the convergence criteria initially and slowly tightens it up as the optimal point is reached.

3.5.3 Reducing Problem Size

Route Elimination

It would not be unusual for the total number of routes summed over all setups and all failure conditions to run into the thousands.(A 5 machine, 10 setup example in chapter 5 generates over 800 possible routes). This number has a direct impact on the work required to reach optimality so we wish to eliminate, from the start, as many routes as possible without jeopardizing the system objective.

One indication that very few routes will actually be chosen by P2 comes from the approximate formulation, P3. For this problem, the number of basic (non-zero) variables cannot exceed the number of constraints, so the number of routes chosen will be low if the number of constraints is low. A medium-size FMS with seven machines and fifteen parts may have at most 15 + 8 * 6 = 63 constraints. The nonlinear formulation P2 can of course choose more routes than problem P2, but it is not likely to require the thousands that are offered. If we knew which routes were not needed, the size of the problem could be reduced substantially.

A good estimate of those routes that are not likely to appear in the optimal solution again comes from problem P2 The "reduced costs" of the final LP can be used to rank all routes according to how much each would increase the cost should the route be used. If the number of constraints in the LP should be fifty, say, then it would not be unreasonable to restrict the number of routes considered in the nonlinear problem to the first 150 in the ranking.

Failure Condition Elimination

The size of the Level 1 formulation is also related to the number of system failure conditions. A ten-machine problem with its 1,024 different conditions would completely overwhelm the computational resource if all had to be considered. However, many of these are of minor importance because the probability of their being occupied is so small. This can be illustrated on a three machine system as follows.

A simple representation of the Condition Transition Diagram that will serve our purposes is obtained by aggregating the conditions according to the number of machines down. This is shown in Figure 3.3.

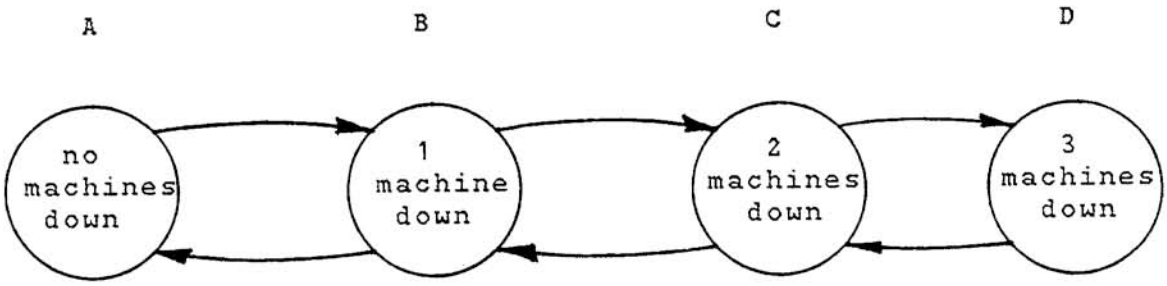


Figure 3.3 Simplified Transition Diagram

Let:

$\lambda e^{-\lambda t}$ = Distribution for time-to-failure.

$\mu e^{-\mu t}$ = Distribution for time-to-repair.

where:

$1/\lambda$ = mean time to failure

$1/\mu$ = mean time to repair

If the mean time-to-failure and repair is 20 hours and 2 hours, respectively, then standard Markov analysis gives the following steady-state probabilities.

PROB(A)	= 0.751
PROB(B)	= 0.225
PROB(C)	= 0.0225
PROB(D)	= 0.000751

The point made here is that most states are not likely to be occupied. The major means we employ to reduce problem complexity is to ignore those conditions that rarely are.

3.5.4 Numerical Example

We apply algorithm for P2 to a three-part , three-machine problem, for two different cases. In the first case, the number of available pixtures for each setup isas not restricted. This is done to validate the procedure by comparing results to those obtained using the approximate formulation P3. In the second case, pixture availability is constrained in order to compare changes relative to the first case. The necessary input problem data is given next.

Production Target:

$N_1 = 75$ $N_2 = 100$ $N_3 = 125$

Number of Resident Setups:

	case 1	case 2
PMAX	15	6

Number of Available Pixtures:

	case 1	case 2
$FMAX_1$	not limited	2
$FMAX_2$	not limited	2
$FMAX_3$	not limited	3

Operation Processing Times:

The setup data, indicating setups, their operations and alternate machines appears in Figure 3.4. Operations are to be performed in the order that they appear.

Failure Model:

A reduced four-state failure model is constructed for this example in order to simplify the computer implementation of the problem. This restriction is artificial and not an inherent limitation of the level 1 formulation. All three machines are assumed to have the same failure and repair

	MACH A	TIME	MACH B	TIME
Setup 1				
Op 1	1	7.94		
Op 2	4	27.82	2	13.11
Op 3	4	12.83	3	9.98
Op 4	3	11.02	2	20.36
Op 5	5	7.94		
Setup 2				
Op 1	1	7.48		
Op 2	3	2.70	4	1.39
Op 3	2	27.58	4	5.14
Op 4	3	19.27	4	22.83
Op 5	5	7.48		
Setup 3				
Op 1	1	7.28		
Op 2	3	7.65	4	11.27
Op 3	2	25.33	3	16.99
Op 4	2	24.88	4	3.18
Op 5	5	7.28		

Figure 3.4 Part Processing Times and Machine Alternatives

characteristics. The distribution satisfied by these failure and repair times is assumed exponential, with a mean time to fail and mean time to repair of 5 hrs each. For the continuous time Markov model shown in Figure 3.5, the steady-state probabilities of being in each failure condition becomes $P^1=.5, P^2=P^3=P^4= .17$.

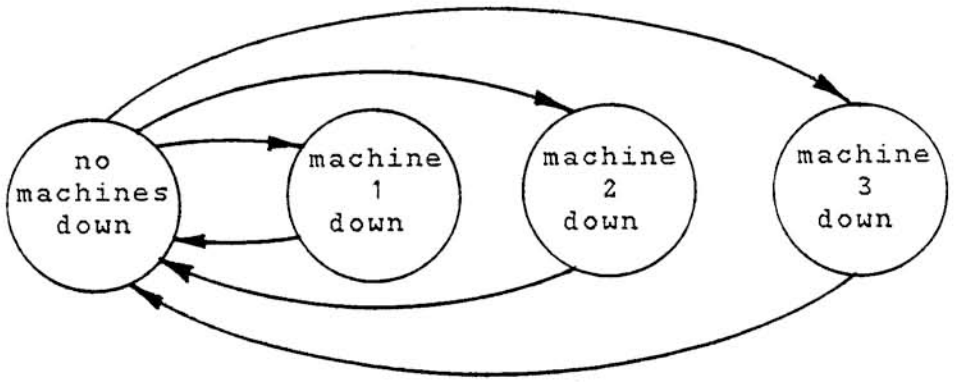


Figure 3.5 Simplified Four-State Failure Model

Results

We are interested in looking at the Level 1 solution for one particular machine failure history. For this run, let us assume that actual machine failures and repairs occur as expected in order to assess, in some sense, the average case. Later, the solution is examined when failures do not occur as expected.

The total number of routes to consider over all failure conditions is 44. It took 11 full iterations of algorithm P2 and .66 minutes of CPU time on an Amdahl computer to obtain the results for case 1. They appear below.

Case 1:

Minimum time,T: Problem P2 = 4310 minutes

Problem P3 = 3950 minutes

Setup Mix:

	CONDITION							
Setup	A		B		C		D	
	P2	P3	P2	P3	P2	P3	P2	P3
1	43	35	0	0	8	11	24	29
2	60	65	30	27	9	8	1	0
3	75	81	28	28	2	0	20	16

Pixture Allocation (Problem P3 only):

	CONDITION			
Setup	A	B	C	D
1	4.4	0	6.9	8.1
2	4.0	7.1	6.2	.5
3	6.6	7.9	1.9	6.4

Figure 3.6 Level 1 Results for Case 1

The table for setup mix states the number of each setup that are to be processed during the total time the system is in each failure condition. Summing over all conditions should yield the production target. The results show good agreement between P2 and P3 formulations but the time to process all setups is 11% higher for problem P2. This is quite consistent with the MVA test results of Section 3.4.3, where reasons were suggested for this behavior. Nevertheless, the good agreement shown for setup mix supports the notion that accurate prediction of relative utilizations, rather than absolute utilizations, is sufficient when balancing workload.

The anticipatory nature of Level 1 is seen through the differences in relative setup mix for each condition. In fact, at times, Level 1 refrains from inputting certain setups altogether. Evidently, it detects more favorable conditions for these parts sometime in the future.

A practical problem in the operation of FMS's is knowing how many of each type of pixture to have on hand for each setup. The advantage of using queueing techniques rather than some procedure that requires simulation is that this information is found directly. The nonintuitive nature of this information is demonstrated in the table for pixture allocation. Notice that although fewer pieces (43) of Setup 1 are allocated to Condition A, Setup 2 needs fewer pixtures (4.0) to achieve its production. Setup 2 must be able to proceed through the system faster than Setup 1 during Condition A.

The results for the constrained case are shown in Figure 3.7. In this case, the completion time must, of course, increase from that of the unconstrained case. Perhaps the most interesting point here is that Level 1 does not fill the system to the pixture limit during each failure condition. With such a strategy, more setups may or may not be processed during a given time period, but it is certain that the completion time of the production target would be delayed.

Case 2: Constrained

Minimum Time, T: 5050 minutes

Setup Mix:

	Condition			
Setup	A	B	C	D
1	50	0	9	16
2	62	28	10	0
3	60	33	2	30

Pixture Allocation:

	CONDITION			
Setup	A	B	C	D
1	2.0	0	2.0	2.0
2	2.0	2.0	2.0	0
3	2.0	3.0	0.4	3.0

Figure 3.7 Level 1 Results for Case 2

Alternate Scheduling Strategies

The effort involved in implementing complex scheduling strategies can be justified only if there is sufficient improvement over other, more elementary, strategies. Two alternatives that should be considered are the following:

1. Selfish- A setup that follows the selfish strategy has no preconceived allocation to resources. The next operation to be performed and the machine that performs it are chosen by simple decision rules. Looking over all those operations that can be performed next for a given setup, the operation and machine are chosen according to which operation can be processed the soonest. (This includes processing time plus queueing time.)

2. Balance - This strategy balances the setup work requirements over all resources for each failure condition. A simple linear program that assumes a saturated system is used in this case. It is similar to the one appearing in Section 3.3 except work is not balanced over the all failure conditions.

For both these strategies, the mix of setups input to the system is the same as that of the production target, regardless of the prevailing failure condition.

We incorporate each strategy into a simulation for Case 1 of our example and obtain the following results. (The Level 1 strategy has taken the name "Defer" to reflect its anticipation of machine failure.)

Strategy	Actual Completion Time	% of Selfish
Selfish	4473 minutes	100%
Balance	4294	96%
Defer	4070	91%

In this example, Level 1 is informed, in advance, of the actual time spent in each failure condition. This information is used to specify the expected failure time that formulation P2 requires. The performance actually achieved by the defer strategy is better than that predicted (4310 minutes, pg.54) because of the pessimistic nature of our network queueing models (see discussion on pg.55). If machines do not fail as expected, then the prediction of the model is not accurate and improvement over other strategies may or may not be realized.

Individual results for a 20 run Monte-Carlo simulation, where the Level 1 formulation is not advised of actual future machine failure, are shown in Figure 3.8.a. The completion time for the production target varies widely depending on the machine failure pattern. Evidently, the running time is not long enough for machines to yield the same "average" failure rates. Also, we see that the completion time, averaged over all 20 runs (4880 minutes), is significantly higher than that expected (4070 minutes). This indicates that applying the expectation operator directly to vect(θ) in equation 3.3 is not strictly valid. We shall attempt, in the next section, to regain some of this loss with an information feedback procedure.

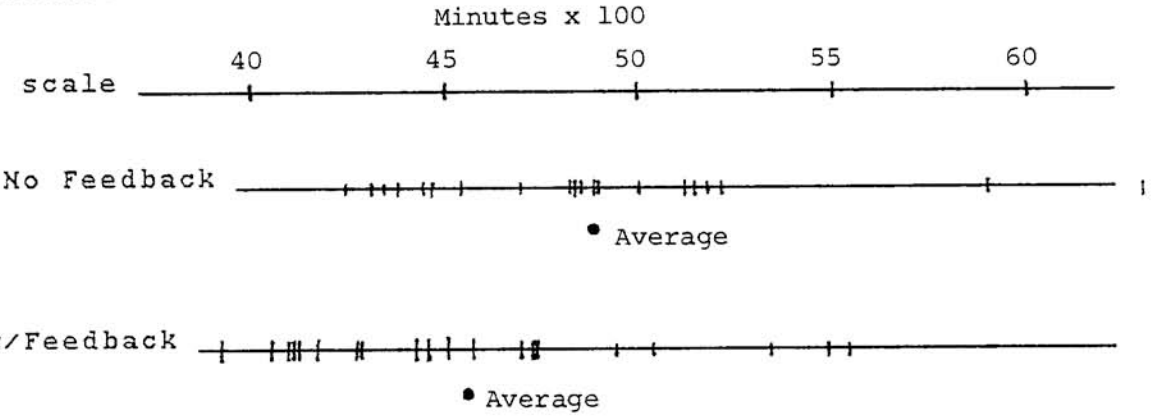


Figure 3.8 Completion Times for 20 Run Monte-Carlo Simulation

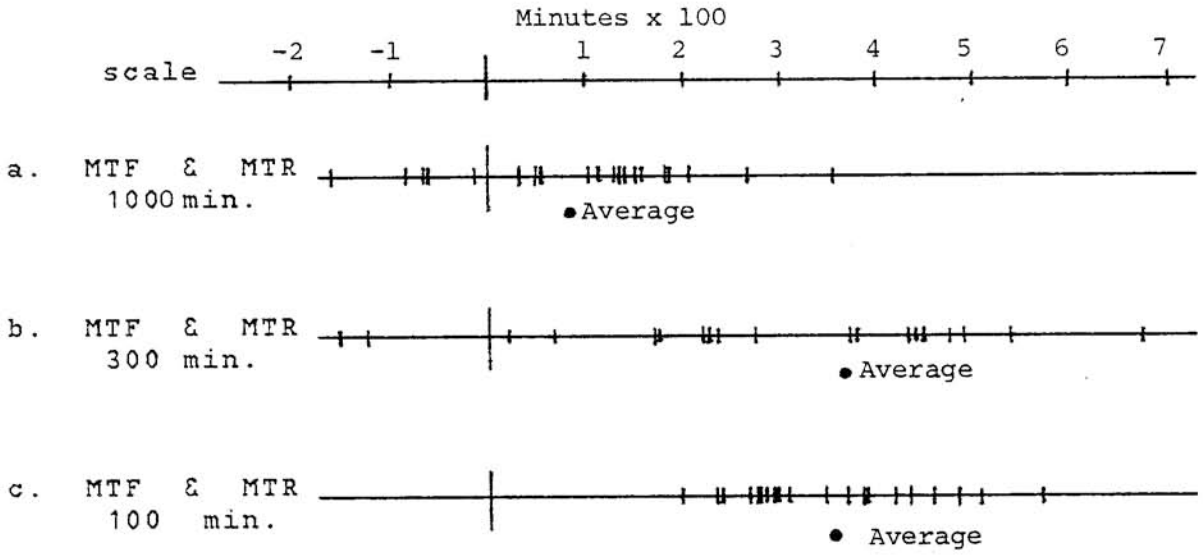


Figure 3.9 Improvement of Defer Strategy Over Balance Strategy

3.6 Information Feedback

The open-loop results of the previous section can be improved substantially if the Level 1 scheduling strategy is periodically recomputed. With every update, a new production target is derived from the original by subtracting those setups that have already been processed. The basic philosophy associated with each update is: "Forget about what has happened in the past. What must now be done to schedule the remaining setups in an optimal manner?" We may view the information feedback process as a method that accounts for past machine failure. Each updated scheduling strategy is a function of the modified production target which depends on the actual failure history. Our implementation anticipates future machine failure through formulation P2 and accounts for past machine failure through information feedback.

A diagram showing the open-loop feedback structure for the Level 1 formulation appears in Figure 3.10.

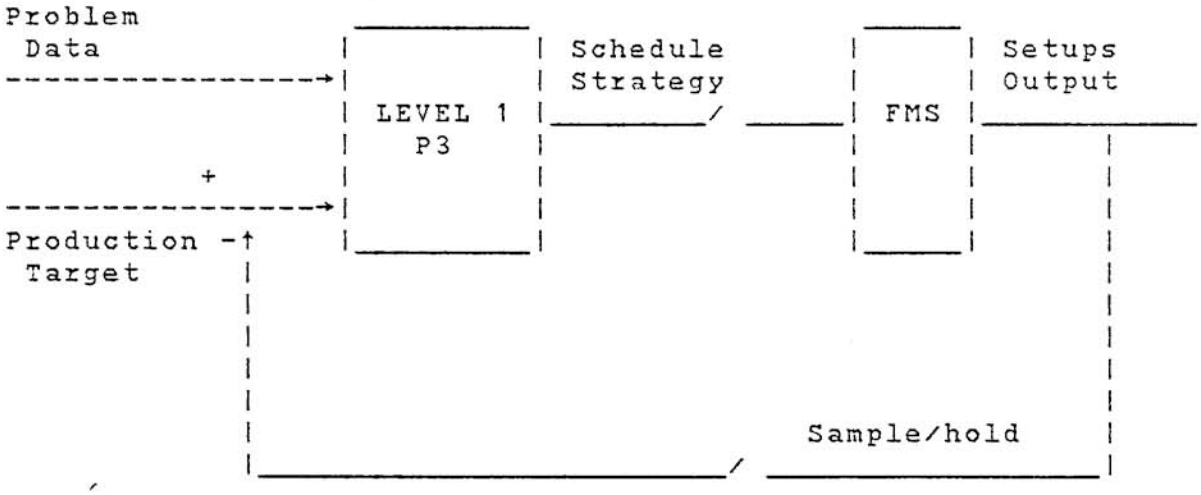


Figure 3.10 Level 1 Open-Loop Feedback

The sample/hold devices indicate that Level 1 will be recomputed periodically rather than continuously. This, of course, goes without saying, since some finite period of time is required to solve P2. However, much longer update intervals (e.g. 1/2 - 1 day) may be adequate. Certainly, if times for future machine failure are known very accurately, (this is the case for planned down-time that is necessary for preventive maintenence), then there is no need to continually re-solve the problem because the solution will not change dramatically. It is only when failure behavior is uncertain that we have to continually hedge our bets with new scheduling strategies.

Example

Production results for the example of the previous section are obtained using the information update implementation of Figure 3.10. New scheduling strategies are obtained every 250 minutes and whenever a machine fails or is repaired. Figure 3.8b shows a general improvement in performance is obtained. The average completion time has droped from 4880 min. to 4552 minutes.

The difference in performance for the Defer and Balance strategy is shown in Figure 3.9 as a function of mean time to fail (MTF) and mean time to repair (MTR). (Both strategies are subject to the same 20 run Monte-Carlo simulation.)

When rates of machine failure and repair are high relative to a given time horizon, predictions of future failure condition occupancy are fairly reliable; that is, the variance in occupation times from run to run is low. The opposite is true when failure and repair rates are high relative to the time horizon of interest. This is reflected in Figure 3.9 where an MTF and MTR of 1000 minutes does not perform as well as that for 100 minutes. In other words, the more certain Level 1 is of future machine failure, the better it will perform.

CHAPTER 4

LEVEL 2: PART INPUT SEQUENCE

LEVEL 3: INTERNAL DECISION RULES

4.1 Introduction

Level 1 succeeded in narrowing the allocation of tasks to resources to a degree. The particular class of systems we consider are not constrained by the material handling system, but rather by the machines. Thus, the Level 1 algorithm focused on resolving those decision variables concerned with machine allocation. These tasks were fully resolved with respect to resources, but the time for each decision was only constrained to lie within periods defined by the occurrence of machine failure conditions.

This chapter develops algorithms that further limit decisions at two final levels. Level 2 narrows the execution time for all the loading tasks and, in so doing, effectively constrains all other tasks as well. This is nothing more than determining a sequence of setups for input to the system. Level 3, being the last step in the decision making process, must finally resolve all decisions fully.

4.2 Level 2: Part Input Sequence

4.2.1 Sequence Subproblems

All of the important resource allocations have been assigned, so our problem is largely one of scheduling in a temporal sense, setups through the system. This is done within the constraints imposed by Level 1. However, the problem is still quite complex due to its size (normally a large number of setups and machines are involved), and because the configuration of the system is changing at random due to machine failure. Since a unique mix of setups (as determined by Level 1) is associated with each failure condition, there is an opportunity for simplifying the problem.

We shall decompose the original scheduling problem into a series of subproblems, one for each failure condition. The results for each condition will be applied whenever the system is in the same condition. The only difficulty is in determining the objective for each interval and the job set over which to construct sequences. Since the length of each time interval is unknown beforehand, this is not completely straightforward.

4.2.2 Subproblem Objective

The Level 1 objective is to minimize the completion time for the production target. This is accomplished in the subproblems by maximizing the rate of production during each instance of a failure condition.

As before, let

λ_{sri} = Production rate of setup s using route r during condition i.

Then the objective for the subproblem associated with each failure condition is:

$$\text{Maximize} \quad \sum_s \sum_r \lambda_{sri} = \lambda_i \quad .$$

s.t.

λ_{sri} satisfies the setup mix for condition i as determined by Level 1.

Even Work Flow

An alternate form for the objective that our algorithm employs is obtained by examining Littles equation for each failure condition:

$$\bar{\lambda}_i = n_i / \bar{\tau}_i \qquad (4.1)$$

where:

$\bar{\lambda}_i$ = Average total setup production rate during condition i.

n_i = Number of setups internal to the system during condition i.

$\bar{\tau}_i$ = Average setup resident time during condition i.

We have assumed in Chapter 3 that the network is closed so the value for n has been determined and fixed. (This determination is another optimization problem that makes the tradeoff between pixture cost and productivity.) With n fixed, minimizing τ is equivalent to maximizing λ. But τ is the sum of average processing time, which cannot be changed by sequencing decisions, and average queueing time which can. So the objective for each subproblem can be realized by minimizing the average queueing time for setups internal to the system.

Queueing time results because events occur that force a resource to accept work at a rate faster than it can handle. Queueing time can be reduced if for every operating hour, a resource is not compelled to do more that an hour's work. Our desire to minimize queueing time can thus be translated to providing each resource with an even flow of work. This criterion, to be formalized in section 4.2.4, will determine the input sequence for setups.

4.2.3 Subproblem Job Set

Finding a deterministic sequence for jobs is not a well posed problem, until a finite job set is specified over which to construct sequences. Since the duration of each failure is not known in advance, the total number of setups to be sequenced is not known either.

In order to proceed, we again make the assumption that equilibrium conditions exist at all times. It is relevant that the optimal control for any problem that reaches equilibrium approaches constant or periodic values. When the control is a sequence of jobs, evidently only those sequences that are periodic need be considered. The following figure shows three sequences of varying periods for three setups whose relative ratios are 1:1:1. The smallest set that contains setups in the desired ratios is called a minimal set. In this case the minimal set is {1 2 3}. Every failure condition is likely to have a different minimal set.

	Sequence	Period
a.	1 2 3 1 2 3 1 2 3 . . .	1 minimal set
b.	1 3 2 3 1 2 1 3 2 3 1 2 . . .	2 minimal sets
c.	2 1 3 2 3 1 3 2 1 2 1 3 2 3 ...	3 minimal sets

Figure 4.1 Periodic Sequences

Since the sequence is periodic, we need only sequence over those jobs contained in one period. This is not much help, though, because the optimal number of minimal sets in a period is not known. However, we are compelled to limit consideration to just one minimal set. In addition to reducing the size of the problem, requirements for the same pixture type are reduced by using one minimal set.

The subproblems are now better defined. The objective is to provide an even flow of work to resources (this is formalized in the next section) and the job set over which to sequence is one minimal set. The sequence found is repeated until the failure condition changes or the setups run out.

4.2.4 Sequencing Algorithm

As stated before, this level will treat the input sequencing decision only. Complexity issues discourage the integration of any other decisions into the process. Operation precedence, delays due to blocking and the hundreds of other things that can alter the journey of a part through the system will not be accounted for in devising a sequencing algorithm.

The flow of work to resources internal to the system will be approximated by the work flow that appears at the entry point (loading station) of the system. We assume that each setup will demand time from every resource it will visit the moment it enters the system. This obviously does not conform to reality but for many systems it is a good first approximation. It is also assumed that the input time instants occur at constant intervals. The length of these intervals need not be specified in our algorithm.

For simplicity, the algorithm is first developed to control work flow to a system with one resource. A minor modification will allow the same development to apply to a system with many resources. Define the following:

J = Finite job set to be sequenced.

$X(t)$ = Subset of J that has entered the system at time t.

t_s = Process time of setup s.

Δ = Constant interval between input instants.

Then:

$$w = \sum_{s \in J} t_s / |J| = \text{average process time per part.}$$

$$W(t) = \sum_{s \in X(t)} t_s = \text{total work load delivered up to time } t.$$

To maximize the average rate of production, we minimize the average queueing time for a part. This is accomplished by feeding the machine work no faster than it can process. The following objective does this by keeping the work delivery rate as close to the average rate as possible. (Formulating an objective that minimizes queueing time directly is possible but cumbersome. The objective we have chosen is effective and easy to understand.)

$$V(k) = \text{minimize over all sequences} \max_{k=0 \rightarrow |J|} |W(k) - wk| \quad (4.2)$$

where time, t, has been replaced by integers, k. The constant increments of time at which setups are input allows this time/integer mapping. Graphically, the objective is shown in Figure 4.2.

Recursion Relation

The cost associated with X(k), that set of jobs that has already entered the system up to time k, is just:

$$C(X(k)) = \max_{m=0 \rightarrow k} |W(m) - wm| \quad (4.3)$$

This may be recursively expressed as follows:

$$C(X(k)) = \max \{|W(k) - wk|, \max_{m=0 \rightarrow k-1} |W(m) - wm|\}$$

$$= \max \{|W(k) - wk|, C(X(k-1))\} \quad (4.4)$$

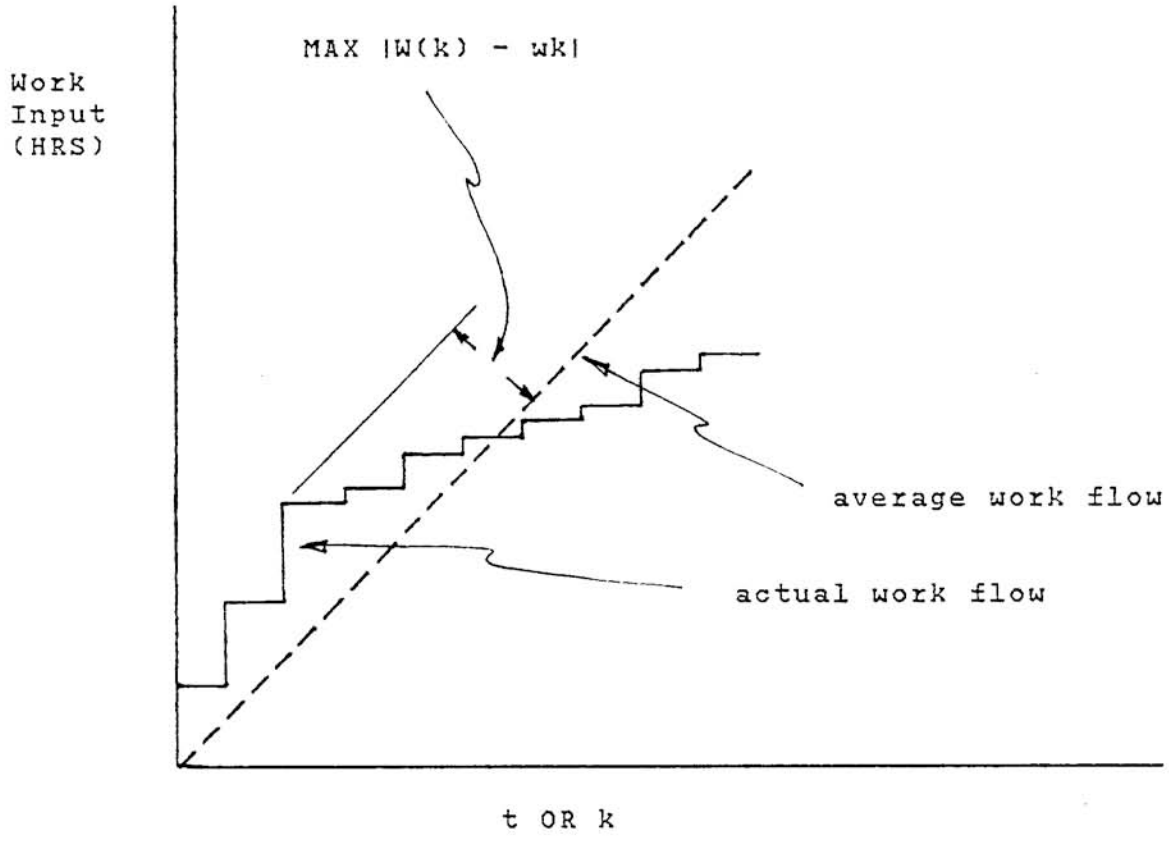


Figure 4.2 Work Flow vs. Time

If we denote α as that job that must be added to X(k-1) to obtain X(k), then the minimum value of C(X(k)) is obtained from the above equation as follows:

$$\min_{\alpha} C(X(k)) = \min_{\alpha} \max \{|W(k)-wk|, C(X(k) - \alpha)\} \qquad (4.5)$$

Since the first argument on the right-hand-side is independent of α, we have:

$$\min C(X(k)) = \max \{|W(k) - wk|, \min_{\alpha} C(X(k)-\alpha)\} \qquad (4.6)$$

This recursive relation defines a D-P algorithm for finding an evenflow sequence. The next section modifies the algorithm so that any number of resources can be treated with no substatntial increase in complexity. For now a simple 1 machine, 4 part example is given to illustrate the algorithm.

Example

The four parts and their process times are:

Part	t_s
1	1
2	2
3	3
4	4

→ w = 2 1/2

The standard D-P diagram for this problem is shown below. The integers within the circles represent the set X(k) and the boxed numbers represent the cost of each node, C(X(k)). The heavy arrows indicate the optimal sequences.

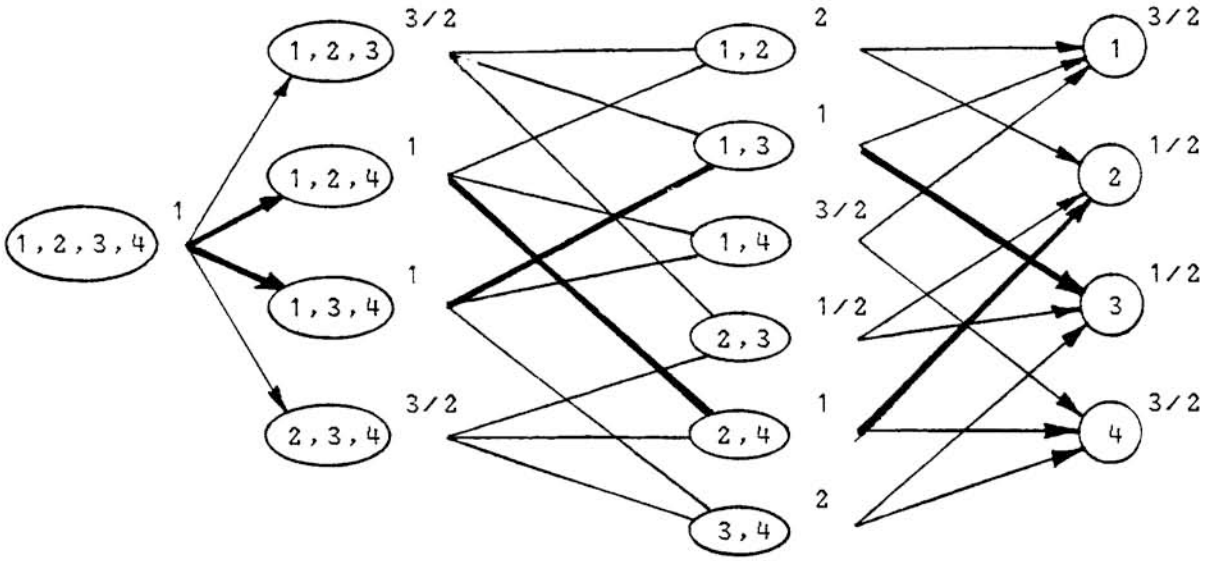


Figure 4.3 D-P Diagram for Even-Flow Sequence

Two sequences satisfy the objective, they are 3, 1, 4, 2 and 2, 4, 1, 3. Note that one sequence is the reverse of the other, a consequence of the objective. This symmetry offers the opportunity of saveing time in obtaining a sequence. We could have stopped the algorithm at the second stage and obtained the same results.

Extension to Multiple Resources

The basic objective (Equation 4.2) can be altered in a number of ways to handle the multiple resource case. It has the form:

$$V(k) = \text{Minimize} \{ \max_{n=0 \rightarrow |J|} F(k) \} \qquad (4.7)$$

One version for F(k) that controls the work flow to all resources is:

$$F(k) = \sum_{i} |W_i(k) - w_{ik}| \qquad (4.8)$$

where i indexes all resources.

4.2.5 Numerical Example

The performance obtained from the even-flow D-P sequencing algorithm is now compared with three other sequencing strategies. They are:

a. Arbitrary - A sequence for the minimal set is arbitrarily chosen and held fixed.

b. SPT - Setups are ranked according to the sum of their processing times on all resources. Those setups with the Shortest Processing Time are input first.

c. Random - The setup entering the system next is chosen at random. In this case, the sequence is not likely to be periodic.

These sequencing algorithms are chosen because they are easy to implement and frequently used. The SPT rule is particularly popular. It has intuitive appeal and is, in fact,

optimal for certain simple scheduling problems. The higher complexity of the even-flow algorithm would prevent its use if any of the alternatives performed as well.

Each of the competing strategies is simulated for the same system used for the example in Chapter 3. Points where this system differs from that of Chapter 3 are the following:

1. Maximum number of setups internal to system = 5.
2. Pixture availability is unconstrained.
3. Machines do not fail.

Also, a larger minimal set is to be processed. It has ten distinct setups and their process times are shown in the table below:

	MACHINE				
SETUP	1	2	3	4	5
1	1 minutes	5	1		1
2	1	10			1
3	1	12			1
4	1	5	4		1
5	1	10	2		1
6	1		15	10	1
7	1		5	20	1
8	1			6	1
9	1		6	8	1
10	1		10		1
TOTAL TIME: (minutes)	10	42	43	44	10

Figure 4.4 Minimal Set for Sequencing Example

The simulation was run for a fixed period of time (4,000 minutes) and the total production noted. The results are:

Stategy	Sequence	Production ** (# Setups)	% of Random
EVEN-FLOW*	8 5 6 4 2 7 1 10 3 9	905(setups)	126%
SPT	1 8 4 9 2 10 3 6 5 7	861	120%
ARBITRARY***		806 MEAN (874 HIGH)	112%
RANDOM		719	100%

* Run Time for Even-Flow Algorithm = 5 seconds (Amdahl)

** Maximum production (saturated system) = 909 setups

*** Statistics were collected over twenty-five separate runs with a different sequence chosen at random for each run.

Discussion

Several points need to be brought out:

1. We have noted that the maximum possible production rate for the example is 909 setup per 4000 minutes (i.e. .23 setups/minute). This rate corresponds to a saturated system where the bottlenecking resource is busy 100% of the time. Any sequence can be made to yield this rate if there is sufficient internal storage for setups. Merely over-saturate the system until the bottlenecking machine is assured a continual supply of work, regardless of the sequence. Sequencing becomes an issue because of the high cost of the storage and the number of pixtures needed to flood the system.

There is further motivation for getting the most production with the least number of resident setups. With larger internal populations comes congestion in various forms. Setups are impeded in their movements about the system and a point is reached where increasing the number of residents actually decreases the production rate. A similar phenomenon occurs for automobile flow on congested highways. As the capacity

of the system is reached, setup flow slows until everything stops. No setup can move off a machine because there is no place to put it and no setup can move into a machine for the same reason. The system is said to be in a state of "lock-up". Computer network specialists have proposed different schemes for preventing lock-up in their systems. One if them is analogous to what we do - limit the number of customers (setups) to some maximum number.

2. The Process time for the minimal set were constructed in a very special way. The work requirements of each setup varies significantly among the resources. Thus if the first five setups were allowed to be resident at the same time, machine 4 would go completely idle. A similar situation occurs for machine 2 when the last five setups are resident. The work patterns in this example are so strong that a reasonable sequence could have been constructed manually. However, the pattern need not be much more complex before a manual solution becomes difficult.

3. The Random strategy did not produce setups in exactly the same ratios the other strategies did, but this alone does not account for its inferior performance. The main reason for this stems from the underlying nature of random sequences. In chapter 5 we examine how production due to a random sequence can be lowered from that of a fixed sequence.

4. In fairness, the SPT rule was altered slightly to take advantage of the obvious imbalance in work requirements between setups 1 - 5 and setups 6 - 10. A short job was taken from the first set followed by a short job from the second set. This adjustment gave fair performance relative to the evenflow strategy. However, any sequence that alternates between these two sets is likely to do well. For example, a Longest Processing Time rule, altered in the same fashion, gives a production of 833 setups. The reasonable performances in both of these cases is due mainly in the adjustment made in the structure of the rule. The adjustment essentially embodies the even-flow concept.

5. An adequate supply of each pixture type is assumed for uninterrupted flow of a fixed sequence. When pixtures are constrained, an entering setup may not have access to its pixture type. They may be occupied by like setups still internal to the system. A good question is: Shall we suspend the input flow rate until the proper pixture becomes available, or stray from the optimal sequence and input a setup for which a pixture

is available? Both strategies help and hinder production in complementary ways. Production is increased by inputting setups, but it can be lost if internal delays build up. An occasional deviation from the optimal sequence is innocent enough but pixtures can be constrained that the question placed above may have to be answered with every entry. This research does not answer this question. In the example of the third chapter, we allowed an off-nominal setup to enter so long as the production mix was satisfied within limits (i.e., the assigned production of no one part was allowed to race ahead of that assigned for others).

4.3 Level 3: Internal Decision Rules

To this point, no decision has yet been fully resolved with respect to time; this is the function of the third and last level in our decision making structure. The complexity of fully resolving the constrained decision variables resulting from Level 2, however, is still quite high if optimality (within the constraints imposed by upper levels on decisions variables) is desired.

We shall pull the problem to within the reach of available computation by adopting a view that is less global. As opposed to solving for all decision variables together in one global formulation, only one decision will be resolved at a time using simple decision rules.

A decision rule specifies a criterion or procedure according to which the decision is resolved. These criteria constitute the objective for decisions. Decision rules generally have an extremely narrow view of the problem so they cannot be related to the global objective very effectively. This is why several are generally tried and the best one chosen. The rules we have chosen to resolve decisions are listed below:

1. Next Operation Rule

Once a setup is internal to the system, the order in which tasks are to be performed must be decided if they are not constrained by precedence relations.

That operation whose machine has the least amount of work queued before it will be chosen. This is called the LWQ rule. If each machine alternative has the same amount queued (for example, both are idle), then the machine whose overall workload is highest will be chosen.

2. Input Time Rule

A setup will enter the system from the load station when internal storage space for the setup is available and the system contains fewer setups than the maximum allowed.

Other rules having to do with the intricasies of the material handling system (which cart, which path from machine to machine, etc.) are not needed in this study because of the simple loop-conveyer MHS that is used. This type most simply satisfies the MHS assumptions specified in Chapter 1.

The detail with which we have treated Level 3 does not compare with that of Level 2 nor Level 1. Rules were based on reasonability rather than extensive analysis and simulation. There are two reasons for this. First, the use of decision rules have been studied extensively in the past twenty-five years, and their success or failure seem to be very problem dependent. Results obtained for the particular system treated here would not necessarily translate to other systems. Second, and more to the point, it is not clear what the payoff would be from a thorough examination of Level 3. If decision variables are properly constrained down through the upper levels (this is indeed the intent), the bottom level should not have the freedom to adversely effect the objective to any great degree.

CHAPTER 5

TRANSIENT PHENOMENA IN THE FMS

5.1 Introduction

The structure for decision making that was developed in Chapter 2 together with the algorithms of Chapters 3 and 4 must be tested on a system of realistic proportions to demonstrate the utility of our approach. In particular, the use of equilibrium assumptions and the organization of decisions is to be evaluated for the class of systems we have chosen.

Before discussing the results of scheduling procedures, certain effects that are inherent to all dyhamic systems must be examined. These effects come under the general heading of "Transient Phenomenon" and arise from disturbances (machine failure) that act on the system. The effects transients have on the system is of interest because our algorithms were developed under the assumption that transients do not exist. In order to assess the vulnerability of our algorithms, the manner in which these phenomenon degrade performance should be understood.

5.2 Transient Phenomena

Three possible mechanisms by which performance can differ from that predicted using equilibrium assumptions are discussed below:

i. Surprised Parts Effect

Our level 1 formulation assumes equilibrium behavior prevails at all times. If a number of setups are allowed into the system according to one strategy , they may not get through the system before the system changes to another and new equilibrium conditions are assumed to prevail. It may be that setups previously allowed in are precisely those that are least preferred by a new strategy. The system will not be "legitimate" until these nuisance setups are able to exit, which may not be for some time.

ii. The Differential Rate Effect

For every failure state, the system processes a mix of parts that is determined by the Level 1 scheduling algorithm. Hence, associated with each failure state, is an equilibrium part flow rate, r° that depends on part processing times and the resident number of parts in the system. The actual flow rate, r, will equal r° during equilibrium conditions only. In general, the equilibrium rates from one condition to another will not be the same and may, in fact, be quite different. This mismatch can create problems.

The effect is most easily seen in a simple FMS with flowshop routing where setups flow from one machine to another in the same order (Figure 5.1).

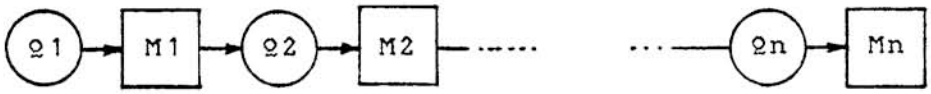


Figure 5.1 Flowshop Routing

Consider what happens when the system changes from State "1" to State "2" and $r^{0}_{1} >> r^{0}_{2}$. Parts resident in the system at the time of the state change will drain from the system faster than entering parts can fill it up. The result is that idle time initially builds up on machines early in the sequence and then moves down the line as parts exit machines faster than they enter. For closed systems, a similar effect occurs for the other situation, $r^{0}_{2} >> r^{0}_{1}$.

iii. The Mini-Differential Rate Effect

System productivity can also be lowered in a way that is related to the second effect, although on a smaller scale. With a change of state comes a change in setup mix and an associated change in setup sequence. When the sequence changes, flow rates to individual machines are altered slightly. If there are not sufficient setups queued before each machine, these fluctuations can precipitate idle time. Figure 5.2 shows a Gantt Chart for a three-machine, three-part problem that has reached deterministic equilibrium by time t_1. (Note that parts need not be queued between

problem that has reached deterministic equilibrium by time t_1. (Note that parts need not be queued between machines at this time. Since all machines finish at once, parts can proceed directly from one machine to another.)

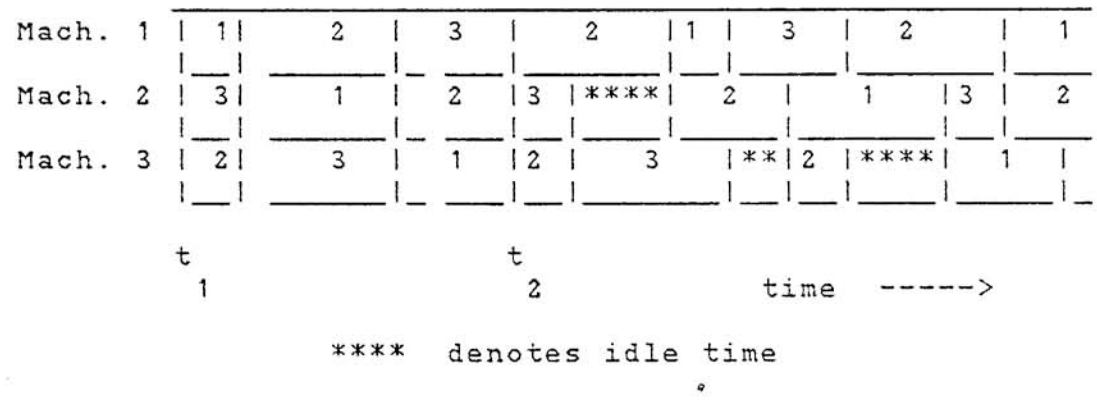


Figure 5.2 Transient Effect iii.

At time t_2, the sequence changes from 1 2 3 to 2 1 3. The result is that the part flow out of machine 2 is faster than the flow rate in and because there was no part queued before it, it must go idle. Eventually, buffers will fill and act as an inventory to isolate each machine from differential rate effects and a deterministic equilibrium reinstates itself. Until that happens, idle time will accrue and lower productivity. If sequence changes are frequent, the loss can be substantial.

This phenomenon is studied for the five-machine system described in Section 5.3. A simulation is run for a fixed length of time for two sequence strategies, Fixed and Random, and performance measures noted. For the random sequence, each part had the same probability of entering next while the fixed strategy chose an arbitrary sequence and held it.

Two different operation precedence relations, judged to generate the extremes in system behavior, are used. The strict precedence relation conforms to flow-shop routing and parallels the example of Figure 5.2. A free precedence relation makes no constraint on the sequence of operations. In this case, the decision rule of Chapter 4 is used to determine the next operation. The results showing throughput and utilization for all four combinations is given in Figure 5.3.

	STRICT PRECEDENCE		FREE PRECEDENCE	
	THROUGHPUT (setups)	AVE. UTIL.	THROUGHPUT (setups)	AVE. UTIL.
FIXED	1932	.933	1980	.955
RANDOM	1633	.793	1823	.883

Number of Resident Parts = 8

Figure 5.3

As expected, for a given precedence relation, the random strategy performs significantly worse than the fixed strategy. The difference in performance is approximately 18% and 9% for the strict and free precedence relations, respectively. Also not unexpected is the superior performance of the free precedence relation over the strict relation. It is well known that freeing operation precedence constraints will increase utilization (Conway,1976). But note in this example, the improvement over a fixed/strict strategy is not very great.

The lesson learned here is: Always try to hold the sequence of parts into the system fixed. It may not be optimal but it is likely to be very good.

The point at which equilibrium assumptions become invalid in a practical sense really cannot be determined without employing simulation methods. In the following section, a numerical study is performed that compares algorithm performance for changes in failure condition of varying duration.

5.3 Algorithm Performance

5.3.1 System Configuration and Part-Set

System Configuration

Our three-level decision structure has been integrated into a simulation of the system shown in Figure 5.4. It has 5 machines with two input buffers each and uses a conveyer MHS (40 ft./min.). The queueing discipline is First-Come-First-Serve and if there is no room in the buffer for an arriving setup, it circulates around the MHS and attempts entry at a later time.

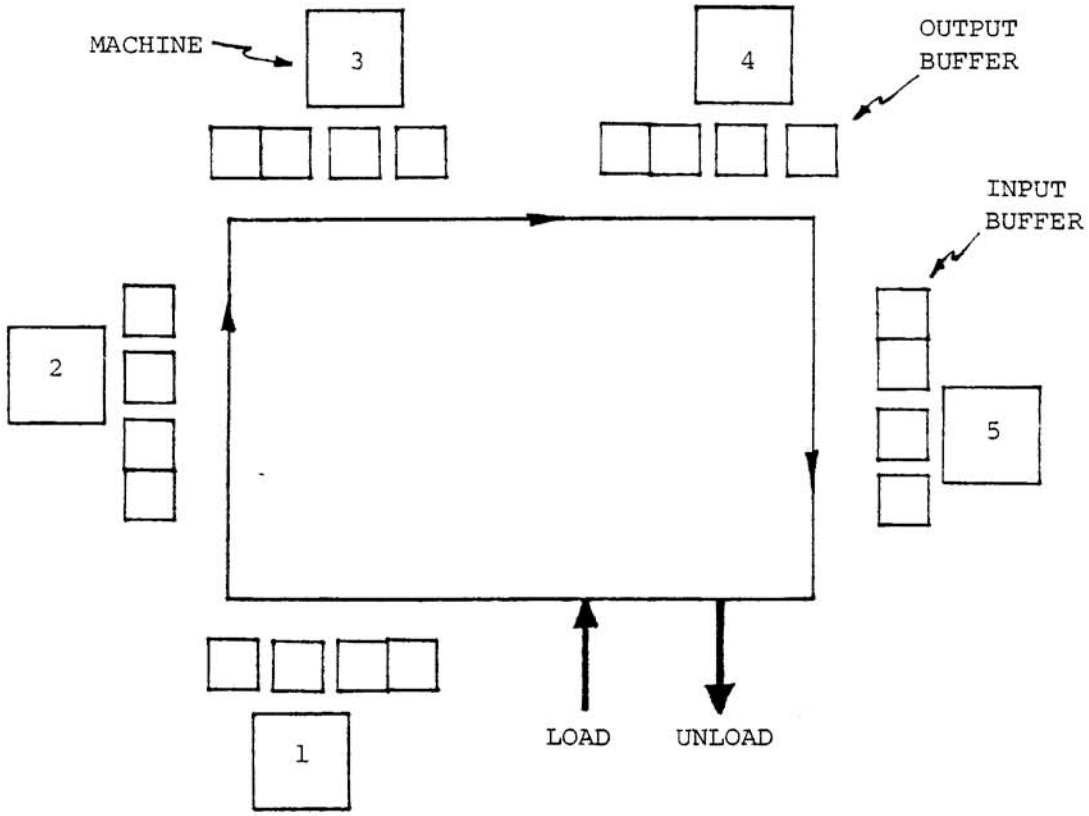


Figure 5.4 System Configuration

Part Set

There is a production requirement for an equal number of 10 different setups. Each setup has five machining operations plus a load and unload operation. Each operation may be performed on one of two machines with process times that are, in general, different. Appendix C gives this information for the ten different setups.

The number of resident parts is chosen to saturate the system (max=15) and the number of pixtures was left unconstrained. This allows the use of the approximation formulation of Section 3.3 which avoids certain complications of implementing the full nonlinear formulation. In this chapter, we are concerned with evaluating a scheduling concept rather than testing the feasibility of an implementation. Thus, restricting ourselves to the linear implementation should not invalidate the conclusions we reach about that scheduling concept.

5.3.2 Loss Due to Transient Phenomena

The algorithms at both Level 1 and Level 2 are designed under the assumption that equilibrium exists for the duration of each failure condition. This is, of course, not true but to what extent are we penalized for assuming it is? Consider the system during one instance of some failure condition and define the following:

$\vec{u}$ = a symbol representing a "control vector" for the problem. It represents all decisions to be made during the period of interest.

$\bar{\lambda}_a(\vec{u})$ = average production rate during the period of interest, assuming actual nonequilibrium conditions. It is a function of the decisions to be made.

$\bar{\lambda}^*_e$ = maximum average production rate assuming equilibrium conditions.

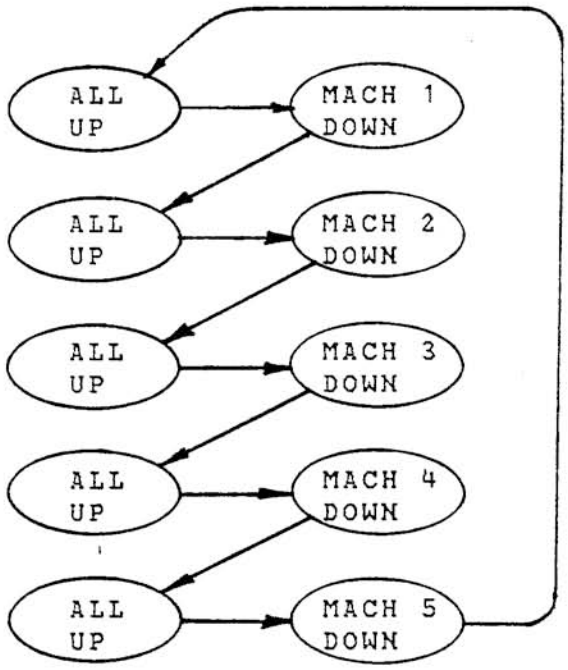
We are interested in the drop in performance due to applying optimal equilibrium controls when the system is not in equilibrium. That is:

$$\text{Performance Degredation} = \bar{\lambda}^*_e - \bar{\lambda}_a(\vec{u}^*_e)$$

where $\vec{u}^*_e$ = optimal control assuming equilibrium conditions.

The degradation represents an upper bound to the performance that can be recovered if nonequilibrium techniques are employed.

The collective loss over all instances of all failure conditions is easily obtained using simulation. Actual algorithm performance is obtained and compared with predicted performance based on equilibrium assumptions. In order to quantify the effect transients have, they must be controlled in some way. This is done through a specially constructed machine failure model. It is most succinctly described by the state transition diagram below.



d = holding time for each failure condition.

Figure 5.5 Failure Condition Transition Diagram

The state alternates between the "all up" condition and one of five "down" conditions and is held in each for the same length of time. As d is decreased the general level of nonequilibrium behavior increases. Algorithm performance is obtained as a function of d.

Experimental Results

We shall vary d relative to a point of reference that is characteristic of the system/part set combination. Define the system response time, σ, as follows:

$$\sigma = E_{sr} \{ \Sigma_{m} \tau_{srm} \} \qquad (5.1)$$

It is the average length of time it takes a part to go through the system. Let:

$$\tau = d/\sigma \qquad (5.2)$$

For high values of τ, system performance should approach that predicted by the Level 1 scheduling strategy.

The simulation is run for a fixed length of time and performance obtained for various values of τ. Once again, both strict and free operation precedence relations are tested. The results appear in Figure 5.6.

Tau	Free Prec. Production (Setups)	Strict Prec. Production (Setups)	% of predicted production* Free	Strict
6.0	179.7	173.5	97	93
3.0	177.2	170.0	95	92
1.4	173.4	162.1	93	87
0.28	170.0	145.5	92	78

* Predicted Production = 185.6 setups

Figure 5.6

The effect τ has on actual production is clearly seen. As τ decreases, production decreases. The level of degredation from equilibrium performance reaches 92% and 78% at τ = .28 for free and strict precedence, respectively. If failures occured at this frequency, this example would experience a sizable loss for the fixed precedence relation. Performance for the strict precedence relation drops off very sharply at τ = 1.5. At this point, the loss of deterministic equilibrium is beginning to be felt. The holding time is so short at τ = 0.28 (d = 10 minutes) that we have, essentially, a random sequence of parts being input (one part input every 2.35 minutes). Notice that the total drop in performance for this example is roughly the same amount exhibited in going from a fixed sequence to a random sequence in Section 5.2 (no failures in that case). This implies that performance is being lost mainly through the second and third transient effect rather than the first.

In the end, only simulation will validate the goodness of one strategy over another. This we do in the next section.

5.3.3 Alternate Scheduling Strategies

We compare the performance obtained by the Level 1 strategy with that of the two others introduced in Section 3.5.4, Selfish and Balance.

The three strategies are compared for different values of τ below.

τ	SELFISH		BALANCE		DEFER	
	Prod. (Setups)	Ave. Util.	Prod.	Ave. Util.	Prod.	Ave. Util.
6.0	131.6	0.996	161.2	0.961	179.7	0.973
1.0	132.0	0.994	157.5	0.943	173.4	0.958
0.28	131.6	0.996	156.4	0.951	170.0	0.959

Figure 5.7

Since the system is assumed to saturate for this example, a good upper bound on the improvement of Defer over Balance is easily computed to be 12%. This is obtained merely by adding up the operation processing times for the two strategies and deriving production rates. The actual improvement runs from 11.5% for high values of τ to 9% for low values. The selfish strategy does a good job at keeping the machine utilized (average utilization over 99%) but the defer strategy out performs it by 33%, averaged over τ. This illustrates the fact that maximizing machines utilization is not necessarily a good objective for scheduling.

The results show that the actual improvement of the defer strategy over the balance strategy is very close to the predicted improvement regardless of the value for τ. An important point is that our strategy may indeed be adversely effected by transients but other strategies may be as well.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

This study has developed an approach to scheduling FMS's that includes features not considered before: machine failure and constraints on pixture type. In order to obtain good performance and remain within a budget imposed on problem complexity, a multi-level solution strategy was developed where each level further refines the decisions passed on from the level above. The framework allows much freedom in exactly how component algorithms are constructed. It enabled us to combine two disparate disciplines in a complementary fashion, queueing theory and scheduling theory. The class of FMS's considered is quite general as is the part set to be processed.

The conclusions reached through this study are enumerated below:

1. A scheduling strategy that anticipates machine failure can be very beneficial, but its success over other, less complex, strategiesis not assured. It is dependent on the part set to be processed and the manner in which machines fail. For any particular case, our strategy must be evaluated along with others using simulation methods.

2. Queueing methods (MVA in particular) have shown they yield good predictions for purposes of balancing work among resources. It is not necessary that they predict system performance exactly. Relative accuracy seems sufficient.

3. The regularity (periodicity) of the setup sequence is very important for systems that must limit the number of resident setups allowed. A random sequence creates minor variations in work flow rate to each machine with the result that idle time can build up. This indicates that it may be advisable to keep extra pixtures of certain types on hand to insure that each setup enters the system uninterrupted, according to its place in the sequence.

4. Setup input sequences that rely on a concept of even-workflow can be constructed and they can be effective. It is not necessary reasonable performance.

Certain limiting assumptions were made in the development in order that a foothold on the problem could be secured. Thus the complicating effects of nonequilibrium, blocking, and internal congestion were not modelled. However, they continue to threaten most systems. Continued effort is required to understand the dynamics of these effects and their influence on system performance. Particular topics that extend the ideas of this research are the following:

1. A very difficult, but important, problem lies in the general area of suboptimal decision making in large problems. Structured guidelines for approaching large problems without any perceivable structure themselves would be most useful.

2. Some systems can reroute setups in the event of machine failure but it is necessary to physically move the associated tools from the failed machine and place them on another. The high change-over time required and limits on the tool capacity of each machine add a new wrinkle to the problem considered in this research. How long should a machine be down before changing tools becomes worthwhile?

3. Consideration of capacity limited material handling systems carries with it myriad questions. How should setups be routed through a crowded system? When do congestion effects begin to degrade performance and how is this detected? How should strategies be altered when portions of the MHS fail? The high interaction between the MHS and machines makes this topic, potentially, very important.

4. The objective of the sequencing algorithm is to feed work to each resource at a rate commensurate with its steady-state average. During transient conditions when the equity in machine backlog can be upset, this strategy may not be appropriate. A minor adjustment in the sequencing algorithm will allow an arbitrary work "profile" to be tracked. Specifying the profile to match actual resource availablitiy is the challenge here.

5. Constructing setup input sequences when pixture constraints prevent an uninterupted flow deserves more attention. Should the input rate be halted until the needed pixture becomes available, or, should one setup be chosen among those with idle pixtures? In the later case which setup is best?

APPENDIX A

ALGORITHM FOR ALTERNATE LEVEL 1 FORMULATION

The form for the approximate formulation is taken from problem P3 (page 35) as follows:

$$\text{Minimize} \quad T$$

$$\vec{x}, T$$

$$\text{s.t.} \quad A_1 \vec{x} \leq N_s$$

$$A_2 \vec{x} \leq P(T)\, T \qquad \vec{x} \geq 0$$

The formualtion is linear except for the term P(T)*T. If the term P(T) is fixed at a constant value, the the problem is completely linear and easily solved.

Consider the simple algorithm on the following page. It is based on the simple idea of successive substitution and is not guaranteed to converge as it stands now. However, it may be successfully modified with the help of the following lemma.

Lemma A.1

Let T* be a solution to P3.

a. If $T^{k+1} < T^k$, Then $T^* \leq T^k$

b. If $T^{k+1} > T^k$, Then $T^* > T^k$

Proof:

First note that P(T)*T is a nondecreasing function of T. This is seen through physical considerations since P(T)*T is the total duration of time a failure state is occupied.

Algorithm A.1

Step 1 : Set k=0 and choose an arbitrary initial starting point for T^k.

Step 2 : Evaluate $P(T^k)$.

Step 3 : Solve the following LP:

$$\underset{x,T}{\text{minimize}} \quad T^{k+1}$$

$$\text{s.t} \quad A_1 \vec{x} \leq N^s$$

$$A_2 \vec{x} \leq P(T^k)\, T^{k+1}$$

$$\vec{x} \geq 0$$

Step 4 : If $|T^{k+1} - T^k| \leq \delta$, then stop. Problem P3 has been solved to within the tolerance specified by δ (δ is chosen to be some small value, e.g., $\delta = .001 * T^k$)

Otherwise, set k=k+1 and go to step 2.

Proof continued:

a.) If $T^{k+1} > T^k$, then $T^* \leq T^k$.

Since T^{k+1}, vect(x) is a feasible solution in step 3 and and $P(T^k) * T^{k+1} > P(T^k) * T^k$, then T^k, vect(x) is feasible solution for problem P3. By definition, $T^* \leq T^k$.

b.) If $T^{k+1} > T^{k}$, then $T^{*} > T^{k}$

Since T^{k+1} is the minimum feasible solution in step 3 and $T^{k+1} > T^{k}$, T^{k} cannot be feasible for problem P3. If it were, the optimality of T^{k+1} in step 3 would be violated.

Therefore, $T^{*} > T^{k}$

***

Lemma A.1 defines a series of upper and lower bounds for T*. If these bounds fail to converge in algorithm A.1, it is quite easy to intervene and generate tighter bounds; merely choose T super(k) to lie somewhere in between the two best bounds so far obtained. This procedure will yield T* to any accuracy desired.

APPENDIX B

MEAN VALUE ANALYSIS

B.1 MVA Algorithm

After a few definitions are stated, the fundamental equations for MVA and the algorithm will be given. Following this, heuristic extensions to the theory will be discussed.

Definitions:

n_s : Population size of setup s. (Eventually, a class will be associated with each component of vect(n). During the MVA development, routes and failure conditions are not distinguished.)

$\vec{n} = (n_1, n_2, \ldots, ns)$: Population vector.

$t_{s,m}$: Mean service time of setup s at machine m.

q_m : Equilibrium mean queue size at machine m.

$q_{s,m}$: Equilibrium mean number of setup s at machine m.

$\tau_{s,m}$: Equilibrium mean waiting time (including service time) of setup s at service center m.

λ_s : Setup s throughput.

e_i : $|S|$-dimensional unit vector in direction i.

M_s : Set of machines setup s visits for processing.

Also, vect(n) may appear as an argument for certain quantities, indicating that they are evaluated for a network with population vect(n), e.g.:

$$q_m(\vec{n})$$

for mean queue size.

Our basic mean-value relation (Reiser and Lavenberg, 1978) stated without proof is:

$$\tau_{s,m}(\vec{n}) = t_{s,m}\,[1 + q_m(\vec{n} - \vec{e}_s)] \qquad (B.1)$$

It relates the waiting time for a network with population vect(n) to the equilibrium average queue size of a network with one less setup. The equation is intuitively very pleasing. It states that the waiting time for a setup is equal to its own service time plus the amount of time required to service the mean backlog of a network with one less setup. In other words, an arriving setup sees a network with itself removed.

It is this remarkable equation that suggests an algorithm that starts with a network having no setups and iteratively increasing the population size until the required level for each setup is reached. The two remaining relationships needed to define the algorithm are:

1. Little's result for classes:

$$\lambda_s = n_s / \Sigma_m \tau_{s,m} \qquad (B.2)$$

2. Little's result for machines:

$$q_{s,m} = \lambda_s \tau_{s,m} \qquad (B.3)$$

The iterative procedure for computing mean waiting times, mean queue size and throughputs is as follows:

Algorithm 1

1. Set $q_{s,m}(0) = 0$ for all m and s.

2. For indices $i_1 = 0,1,\ldots,n_1;\ldots;$ is $= 0,1,\ldots,ns$, loop through steps (3) to (5) while changing i_1 most rapidly.

3. $$\tau_{s,m} \leftarrow t_{s,m}\left[1 + \sum_j q_{j,m}(\vec{i-e_s})\right] \qquad (B.4)$$

4. $$\lambda_s \leftarrow i_s / \sum_m \tau_{s,m} \qquad (B.5)$$

5. $$\vec{q_{s,m}(i)} \leftarrow \lambda_s \tau_{s,m} \qquad (B.6)$$

There are $n_0 * n_1 * \ldots * ns$ recursive steps in the algorithm above and it can be manipulated such that at each step the number of additions and multiplication/divisions is bounded by $4*|S|*|M|$. M is the set of all machines.

It is evident from this operation count that the computational burden associated with MVA, as developed so far, is much too high for practical puposes. In Section B.3, we will present a modification to the method that speeds up computation but yields approximate results.

B.2 An Extension to MVA

The assumption that, at a given machine, all setups take their service time from the same exponential distribution, is too constraining for the realities of FMS's. To allow our methods to more accurately render setup-dependent service distributions, we make the following heuristic modification (originally suggested by Reiser) to the mean value Equation B.7

$$r_{s,m}(\vec{n}) = t_{s,m} + \sum_j t_{j,m}\, q_{j,m}(\vec{n-e_s}) \qquad (B.7)$$

Notice that the average time each class spends at machine m, t_{SM}, has been distinguished and brought within the summation. This maneuver does not comply with the original MVA assumptions. Numerical results show,however, that MVA, using the approximation of the next section and extension above, still predicts system performance with reasonable accuracy. These results are given in section 3.4.3.

B.3 An Approximation to MVA

We wish to wrest from Algorithm B.1 the need to iterate through all various populations before arriving at the final level $n_1, n_2, \ldots n_S$. Let us write:

$$q_{s,m}(\vec{n-e_j}) = q_{s,m}(\vec{n}) - \epsilon_{s,m}^{j}(\vec{n}) \qquad (B.8)$$

where:

$$\sum_m \epsilon_{s,m}^{j}(\vec{n}) = \delta_{s,j} \; ; \; \delta_{s,j} = 1 \text{ if } s = j \qquad (B.9)$$

0 otherwise

If the correction terms:

$$\epsilon_{s,m}^{j}$$

can be found (or estimated) from the queues for a network with population vect(n):

$$q_{s,m}(\vec{n})$$

then the recursion of Algorithm B.1 can be replaced by a simple iterative algorithm. Combining both the extension and approximation, Algorithm B.1 becomes:

Algorithm B.2

1. Initialization: $q_{s,m} = n_s / |M_s|$ for all m in M_s,
   $q_{s,m} = 0$ for all m not in M_s.

2. Repeat steps (3) to (6) until a suitable convergence criterion is met.

3. $\epsilon_{s,m}^{j} = f(q_{s,m})$. (B.10)

4. $\tau_{s,m} = t_{s,m} + \sum_j t_{j,m} (q_{j,m} - \epsilon_{j,m}^{s})$ (B.11)

5. $\lambda_s = n_s / \sum_m \tau_{s,m}$ (B.12)

6. $q_{s,m} = \lambda_s \tau_{s,m}$ (B.13)

One estimate for the correction terms that is fairly intuitive, easily evaluated and gives good results is the following:

$$\epsilon^{j}_{s,m} = \frac{q^{\rightarrow (n)}_{j,m}}{\sum_i q^{\rightarrow (n)}_{j,i}} \qquad s = j \qquad \text{(B.14)}$$

$$0 \quad \text{otherwise.}$$

This expression attempts to approximate the difference in queue length an arriving setup would see between two networks, one network having one less setup of class s. Viewed another way, the approximation claims that if one setup of class s were to be added to the network, its mean queue length at each machine would increase in proportion to the queue lengths for the network without the additional setup. Queue lengths for all other setup classes would remain unchanged.

The operation count per iteration cycle for this algorithm can be manipulated to $9*|S|*|M| + |S|$. Experience has shown that the number of iterations required such that the change from one iteration to the next is less than 1% is on the order S/2 when the procedure is started from an arbitrary point. When given a good start, far fewer iterations are needed. This will be significant in the algorithms to follow. The complexity of Algorithm B.2, then, is of the order $O(S^2)$ which compares very favorably with Algorithm B.1.

Algorithm B.2 has another major advantage over Algorithm B.1. The population of each class is no longer constrained to be integer, or even greater than one for that matter. This enables us to specify an average population size for each class that is fractional. Since the average population of individual classes for an optimal solution is not expected to be integer, this feature is most welcome.

Convergence

Steps (3)-(6) of Algorithm 2 can be combined into the following nonlinear interative form for n:

$$q^{k+1}_{s,m} = \frac{n^{k}_{s}\,\tau_{s,m}}{\sum_{i} \tau_{s,i}} \qquad \text{(B.15)}$$

where:

$$\tau^{k}_{s,m} = t_{s,m} + \sum_{j} t_{j,m}\, q^{k}_{j,m} - t_{s,m}\, \frac{q^{k}_{s,m}}{\sum_{i} q^{k}_{s,i}} \qquad \text{(B.16)}$$

Convergence has not been analytically shown for any but the most trivial cases. However, in no instance has the algorithm been found not to converge.

The criterion used in Algorithm B.2 is as follows. Iterate until:

$$\max_{s,m} \frac{q^{k+1}_{s,m} - q^{k}_{s,m}}{q^{k}_{s,m}} \leq d \qquad \text{(B.17)}$$

APPENDIX C

Part Processing Times for Chapter 5 Example

	Machine	Time	Machine	Time
Part 1				
op 1	6	4.80	2	2.33
op 2	6	2.29	4	1.81
op 3	5	1.98	2	3.55
op 4	6	0.37	3	3.45
op 5	2	1.00	3	4.45
Part 2				
op 1	6	2.99	3	3.32
op 2	5	4.98	6	2.71
op 3	3	2.98	2	3.20
op 4	6	0.67	2	3.23
op 5	6	1.16	4	4.16
Part 3				
op 1	5	1.51	4	0.50
op 2	2	0.52	5	1.45
op 3	2	1.59	6	2.07
op 4	3	1.27	2	4.55
op 5	2	3.94	6	4.26
Part 4				
op 1	2	4.53	4	1.27
op 2	4	3.27	2	3.71
op 3	3	3.80	6	3.58
op 4	2	2.16	4	4.45
op 5	4	2.49	2	1.63
Part 5				
op 1	3	1.09	5	1.48
op 2	5	3.89	6	0.65
op 3	6	1.62	5	2.52
op 4	4	3.92	2	3.06
op 5	3	3.60	2	2.03
Part 6				
op 1	6	4.68	2	1.36
op 2	4	4.27	6	3.80
op 3	3	3.25	4	0.84
op 4	5	0.70	6	2.46
op 5	3	4.92	4	2.97

	machine	time	machine	time
Part 7				
op 1	5	2.19	6	4.95
op 2	4	2.27	6	3.86
op 3	5	4.85	4	1.81
op 4	6	3.83	5	2.28
op 5	5	4.19	4	0.44
Part 8				
op 1	5	2.86	6	2.89
op 2	3	3.22	6	4.18
op 3	3	3.68	2	2.07
op 4	6	3.19	2	1.48
op 5	4	0.63	6	1.08
Part 9				
op 1	6.	2.78	3	1.02
op 2	5.	4.20	6.	3.94
op 3	2.	4.32	4.	1.79
op 4	3.	4.01	6.	1.32
op 5	5.	1.47	4.	2.13
part 10				
op 1	6.	3.50	5.	1.97
op 2	6.	2.68	2.	4.85
op 3	3.	3.45	4.	3.02
op 4	6.	4.83	4.	1.17
op 5	2.	2.17	3.	1.15

REFERENCES

1. K. R. Baker, "Introduction to Sequencing and Scheduling", Wiley, 1974.

2. Y. Bard, "Some Extensions to Multiclass Queueing", G320-2124, IBM Cambridge Scientific Center, Cambridge, Massachusetts, November, 1978.

3. F. Baskett, K. M. Chandy, R. R. Muntz, and F. Palacios-Gomez, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", J. ACM, Volume 22, No. 2, April 1975, pages 248-260.

4. E. C. Coffman, Jr., "Computer and Job-Shop Scheduling Theory:, John Wiley, 1976.

5. R. W. Conway, W. L. Maxwell, L. W. Miller, "Theory of Scheduling", Addision Wesley, 1967.

6. S. E. Dreyfus, A. M. Law, "The Art and Theory of Dynamic Programming", Academic Press, 1977.

7. W. J. Gordon and G. F. Newell, "Closed Queueing Networks with Exponential Servers", Operations Research, Volume 15, No. 2, April 1967, pages 244-265.

8. G. K. Hutchinson, "The Control of Flexible Manufacturing Systems: Required Information and Algorithm Structures", IFAC Symposium on Information-Control Problems in Manufacturing Technology, Tokyo, 1977.

9. J. R. Jackson, "Jobshop-like Queueing Systems", Management Science, Volume 10, No. 1, October 1963, pages 131-142.

10. J. Kimimia and S. B. Gershwin, "Network Flow Optimization in Flexible Manufacturing Systems", Proc. CDC, January 1979.

11. L. Kleinrock, "Queueing Systems - Volume I", Wiley, 1976.

12. M. Reiser and S. S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks", IBM Research Report RC-7023, Yorktown Heights, New York, March 1978 (to appear in J. ACM).

13. G. Secco-Suardo, "Optimization of a Closed Network of Queues", Complex Materials Handling Systems, Volume 3, MIT Electronic Systems Laboratory Report, ESL-FR-834-5, 1978.

14. J. J. Solberg, "Quantitative Design Tools for Computerized Manufacturing Systems", Proc. Sixth North American Metalworking Research Committee, Gainsville, Florida, April 1978.

15. K. E. Stecke and J. J. Solberg, "Scheduling of Operations in a Computerized Manufacturing System", Optimal Planning of Computerized Manufacturing Systems, Report No. 10, School of Industrial Engineering, Purdue University, December 1977.

16. R. Suri, "Why Simple Queueing Network Models Give Good Predictions for Practical Systems", to appear in Proc. International Conference on Cybernetics and Society, Cambridge, Massachusetts, October 1980.

17. R. Suri, "Resource Management in Large Systems", Ph.D. Thesis, Harvard, Division of Applied Sciences, December 1979, Technical Report 671.

BIOGRAPHY

Richard Hildebrant was born in Aberdeen, South Dakota on August 13, 1948. He attended Broome Community College in Binghamton, New York for the first two years of undergraduate studies and then proceeded to the State University of New York at Buffalo where he received the BS degree, with honors, in Engineering in June 1970. While there, he was inducted into the honor fraternity, Tau Beta Phi.

In September, 1970, he entered graduate school at MIT in the Department of Aeronautics and Astronautics, receiving the SM degree in June 1972. During this time he held a research assistantship with the Measurement Systems Laboratory of MIT and worked in the area of Inertial Navigation and Modern Control Theory. Upon graduation, he took a position as Staff Engineer at the C. S. Draper Laboratory.

Mr. Hildebrant re-entered MIT in 1974 and is now at the Draper Laboratory, continuing his work in Large-Scale System methodology as it applies to manufacturing systems.