R-1385-PR
June 1974

An Experimental Investigation of Priority Dispatching in Aircraft Maintenance, Using a Simplified Model

L. W. Miller, A. S. Ginsberg, and W. L. Maxwell

A Report prepared for

UNITED STATES AIR FORCE PROJECT RAND

Rand
SANTA MONICA, CA. 90406

The research described in this Report was sponsored by the United States Air Force under Contract No. F44620-73-C-0011 — Monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Reports of The Rand Corporation do not necessarily reflect the opinions or policies of the sponsors of Rand research.

R-1385-PR
June 1974

An Experimental Investigation of Priority Dispatching in Aircraft Maintenance, Using a Simplified Model

L. W. Miller, A. S. Ginsberg, and W. L. Maxwell

A Report prepared for

UNITED STATES AIR FORCE PROJECT RAND

Rand
SANTA MONICA, CA. 90406

Published by The Rand Corporation

PREFACE

Over the past fifteen years a great deal of research has been directed toward developing effective dispatching procedures in job shops. Most of that research addresses the processing of jobs that are independent of one another. A considerable portion of the body of knowledge that has accumulated over this period is applicable to a variety of Air Force scheduling problems. Still, many Air Force maintenance organizations work on aircraft and other complex end items and components which, although they consist of independently processable jobs, cannot be returned to service until all associated jobs are completed.

It is the latter Air Force maintenance environment which provided motivation for the present study. Although the representation of the real world in this study is quite abstract, an understanding of the factors that lead to effective scheduling in an abstract environment should contribute to the development of improved scheduling at real bases and depots.

The Air Force's Study of the Automation of the Logistics System at Base Level (STALOG), the recent Maintenance Management Information and Control System (MMICS) field test at K. I. Sawyer AFB, Project Early Bird [an ongoing field test at Ogden Air Materiel Area (AFLC)], as well as some recent component repair scheduling research conducted by members of the AFLC in support of the Advanced Logistics System--all highlight the need for more basic understanding of job scheduling for complex products.

The outcomes from this study add insights to this job scheduling area. To Air Staff, Air Force Logistics Command, and the Air Force Data Systems Design Center personnel who sponsor or conduct studies in job scheduling the study will supply special background for establishing policies and rules, and the study outcomes should be useful in machine-aided scheduling systems.

SUMMARY

This report describes an experimental investigation by computer simulation of a simple assembly shop. Products, which could be aircraft requiring unscheduled maintenance at a base, arrive according to a random process. Each product is composed of a random number of jobs, with each job requiring processing by a single specific type of machine (skill category). The shop consists of a set of machine groups to which the jobs are directed. The individual jobs may be worked upon independently, and a product is considered finished as soon as all of its jobs have been completed. The situation is a simplification of that studied previously by Maxwell in that his jobs contained random numbers of operations that were routed through a network of queues.

The goal of this study was to understand better the roles of various factors in constructing dispatching procedures designed to minimize the average time to product completion. Good dispatching rules must consider processing time in order to reduce waiting in the machine queues. But they must also be able to coordinate the progress of several jobs in a product. This can be done by giving all jobs within a single product a common priority value. In addition, effective dispatching procedures must be dynamic in that they reflect current status whenever a dispatching decision is required.

The rules tested may be classified by three types of choices: (a) by job length, number of jobs, or measure of total product processing requirements; (b) by subset with respect to status of jobs in a product; and (c) by current resource commitments.

It turned out that there is a dependence among the choices; i.e., the best choice of the set of jobs within a product to be considered varies with the attribute selected in (a) above. The best procedure among those tried was an "unstarted work content" rule. That is, give highest priority to the set of waiting jobs whose product has the least sum of processing times.

The statistical significance of the experimental results is discussed and a concluding section entitled "Relationship to Aircraft Maintenance" relates this study to the real world problem that motivated it.

CONTENTS

I. INTRODUCTION

The study of dispatching rules in a job shop context by means of digital computer simulation has had a relatively long history. In most of these studies the jobs to be processed were independent in that they did not combine at any stage of the processing. This model of independent jobs, though representative of many real world situations, does not represent the frequently encountered situation where jobs are assembled into products.

Investigations of job shops with assembly constraints have been reported by Maxwell and Mehra [1968] and by Maxwell [1969]. A summary of the latter appears in Conway, Maxwell, and Miller [1967]. The Maxwell study extended the work of Conway [1964, 1965], also summarized in Conway, Maxwell, and Miller [1967] to include assembly constraints. Siegel [1971] is a more recent study on the same general topic.

Conway investigated a job shop consisting of a network of nine machines. Each job was a sequence of operations. Routings were generated randomly, processing times were independently sampled from an exponential distribution, and interarrival intervals were exponentially distributed. Conway's effort was mainly concerned with evaluating various dispatching rules with respect to flow time and lateness measures.

His dispatching rules were implemented by computing a priority value for each job in the queue of a machine whenever a machine completes an operation. The job with the smallest priority value was selected for processing next. Routings and processing times, but not future arrivals, were available for use in computing priorities.

The most significant result of Conway's study was the superiority of the SPT (Shortest Processing Time) rule--let the priority value of a job be determined by its processing time for the imminent operation; giving precedence to the short jobs minimizes product flow-through time.

Maxwell [1969] added one level of assembly to this structure. The jobs were released to the shop in groups of 2, 5, or 10 jobs, each

group representing a product. Completed jobs went to an assembly shop where they awaited completion of all other jobs in their products. A product left the assembly shop as soon as all its jobs were finished; the assembly operation took no time.

Maxwell and Mehra [1968] investigated priority dispatching with products consisting of "symmetric tree structures" of operations. Operations were arranged in levels, with operations on a given level all having the same given number of immediate predecessors on the prior level. The number of levels was uniformly distributed between 2 and 5, and the number of immediately preceding operations for one operation at any level was uniformly distributed between 2 and 4, with this attribute common to all nodes at the level. A product could consist of, at most, 40 operations. The rules tested were fairly complex, and the authors draw interesting distinctions among classes of rules based on their information requirements.

The study described in this paper is a step backwards in that it deals with simpler product structures than those used by Maxwell. In fact, a modified version of his simulation model was employed. As in the earlier work, there is just one level of assembly, but each job in the product consists of only one operation. On the other hand, each machine group has two machines. It was hoped that additional insight might be gained from working with a simpler situation.

The specific application to which this study was addressed was unscheduled maintenance on aircraft. Typically, a set of malfunctions is reported on an airplane, and repairmen (machines in the model) are dispatched to correct the malfunctions. The model, however, represents a highly abstracted view of the real world. Frequently teams of men are dispatched for one job, there may be concurrency constraints, information on repair times may be poor, and additional jobs can be discovered while a product is in process.

This study is primarily concerned with minimizing product flow times. Although enforcement of due dates was not studied in depth, measurements on tardiness based on a constant due date allowance were made. But none of the priority assignment procedures was specifically designed with tardiness measures in mind, and none used due dates.

II. THE MODEL AND EXPERIMENTAL CONDITIONS

THE MODEL

The job shop consists of a set of *machines* clustered in *machine groups*. Machines within a group are identical with respect to their processing capability and performance. Each *job* is only one operation, to be performed by one machine. The attributes of an operation are the identity of the *machine group* required and its *processing time*. With a *product* are associated a set of jobs, all arriving simultaneously; these must be finished before the product is considered completed. The *assembly shop* is a fictitious shop in which we imagine that jobs completed on a product wait until the last job on that product is completed.

Machines are continuously available, and there are no explicit setup times. Jobs within a product are independent in that there are no sequencing or concurrency requirements. Preemption, lot splitting (i.e., allowing more than one machine to work on a single job), and substitution of machines are not allowed. Processing times are known with certainty but no information regarding future arrivals is available.

EXPERIMENTAL CONDITIONS

The job shop consisted of eight machine groups, each with two identical machines. Interarrival times between products were exponentially distributed with a mean interarrival time of 0.0441 day, or an average of 22.67 products per day. Processing times were exponentially distributed with a mean of 0.1 day. The input tape from which the jobs were read had been set up to allow for a shop having up to twenty machine groups, to provide for runs with a shop of this size. The number of jobs in a product was sampled from a discrete triangular distribution ranging from 5 to 25 with a mean of 15. Then the jobs were randomly assigned to machine groups with equal probability over all twenty groups. (Thus it was possible for a product to have several jobs in a single machine group.) For the eight-machine group shop,

jobs assigned to groups nine through twenty were discarded. In the event that all jobs in a product were discarded, the first processing time in the jobset of the product was randomly assigned to a legal machine group, providing a product with only one job. This procedure resulted in a slightly right-skewed distribution of jobs per product, having a mean of 6.004 and a variance of 42.80.

The nominal utilization was 85 percent. This is the expected amount of work (in days) arriving per day:

$$22.67 \text{ products/day} \times 6.004 \text{ jobs/product} \times 0.1 \text{ day/job} = 13.61 \text{ days-of-work/day}$$

divided by 16 machine days of capacity per day.

Upon arrival, each product was assigned a due date equal to the product arrival time plus an allowance of 2/3 day.

RUN LENGTH

Each rule was tested against the same set of products. The first ten products were loaded into the shop as an initial load, and the next 390 products were used for run-in. The sample consisted of 1800 products (products 401 through 2200), and up to 200 additional products were used for run-out. At most, one or two of the 1800 sample products were left in the system by the time product 2400 arrived.

A PRELIMINARY RUN

Most of the results reported in this paper were carried out under the conditions described above. But four assignment procedures, two of which were not repeated later, were tried in an early series of experiments with the following differences in conditions: There were twenty machine groups, the product arrival rate was 20 per day, and the triangular distribution of jobs per product described above was used. This gave a nominal utilization of 75 percent. The sample size was 5000 products.

III. PRIORITY ASSIGNMENT PROCEDURES

Each time a machine having a non-empty queue becomes available, a priority value, $Z_j(t)$, for each job in that queue is computed, and the job with the smallest value is selected. Variations in assignment procedures, or rules, are achieved by modifying a subroutine called SELECT,* occasionally with other modifications to increase efficiency.

NOTATION

The following notation will be used in describing the priority rules: Let

t = Time at which selection for machine assignment is to be made.

j = Index over jobs.

$p(j)$ = Product to which job j belongs.

$m(j)$ = Machine group required by job j.

$Z_j(t)$ = Priority value of job j at time t.

P_j = Processing time required by job j.

$A_{p(j)}$ = Set of jobs in $p(j)$.

$C_{p(j)}(t)$ = Set of completed jobs in $p(j)$ at time t.

$S_{p(j)}(t)$ = Set of jobs in $p(j)$ that have been started but not completed as of time t.

$UC_{p(j)}(t) = A_{p(j)} - C_{p(j)}(t)$, the set of uncompleted jobs in $p(j)$ at time t.

$US_{p(j)}(t) = UC_{p(j)}(t) - S_{p(j)}(t)$, the set of jobs in $p(j)$ that have not been started as of time t.

$E_j(t)$ = Completion date† of job j if each job in $US_{p(j)}(t)$ were to be started as soon as an appropriate machine became available. A detailed explanation of how this is computed is given below.

---

*The program was written in SIMSCRIPT I [Markowitz, Hausner, and Karr, 1962].

†The term "date" will frequently be used to distinguish between points in time and time intervals.

$Q_{m(j)}$ = Set of jobs in queue at machine group m(j).

$n\{X\}$ = Number of jobs in the set X .

R_j = Release (arrival) date of job j (and all jobs in $A_{p(j)}$).

These definitions are illustrated in Figs. 1 and 2, where we suppose that a priority value is being computed for job 1 belonging to a product with seven jobs with $A_{p(1)} = \{1, 2, \ldots, 7\}$. Job 7 has been completed, so $C_{p(1)}(t) = \{7\}$; jobs 5 and 6 are being processed, so $S_{p(1)}(t) = \{5, 6\}$. Then $UC_{p(1)}(t) = \{1, 2, \ldots, 6\}$, and $US_{p(1)}(t) = \{1, 2, 3, 4\}$.

The computation of $E_j(t)$ is complicated by the fact that more than one job in $US_{p(j)}(t)$ may be in $Q_{m(j)}$. Figure 2 illustrates how $E_j(t)$, (for j = 1, 2, 3), would be calculated when all three jobs are in the same queue. Suppose that there are two machines in the group, and they are working on jobs that will be finished at times T_1 and T_2. Trial assignments are made, always placing the longest job on the first available machine, updating the T's as each assignment is made. This method of tentatively placing jobs on machines was chosen because it minimizes the maximum $E_j(t)$ when there are no more than two jobs involved (that is, in the intersection of $US_{p(j)}(t)$ and $Q_{m(j)}$).*

Figure 3 shows an earliest completion time profile, which is simply the jobs placed on a time scale with the right ends at their values of E . The shaded portions of the figure indicate prior commitments of the machines.

The E's and P's can be used in similar ways, although computing the E values is time-consuming and requires keeping track of completion times for all busy machines. They are potentially useful, however, to avoid starting a job when some other job on the same product cannot possibly be finished for a very long time.

Most of the rules examined in this study are related either to Fig. 1 or to Fig. 3. That is, what are reasonable measures to use in

---

*With $T_1 = 4$, $T_2 = 2$, as in Fig. 2, but with $P_1 = 6$, $P_2 = 4$, and $P_3 = 3$, it would be better to assign job 1 to machine 1 and the others to machine 2.

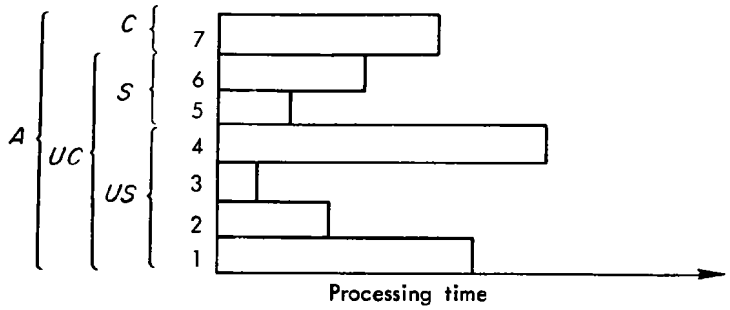


Fig. 1 — Work profile of product composed of jobs 1 through 7

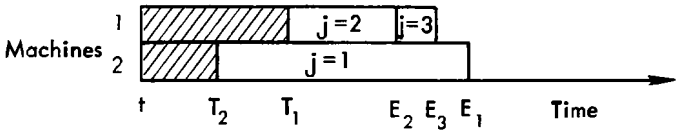
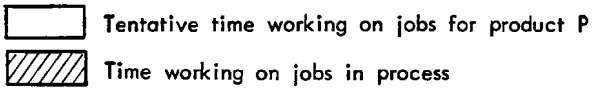


Fig. 2 — Computation of E when one product has three jobs in one machine group. Here $P_1 = 7$, $P_2 = 3$, and $P_3 = 1$



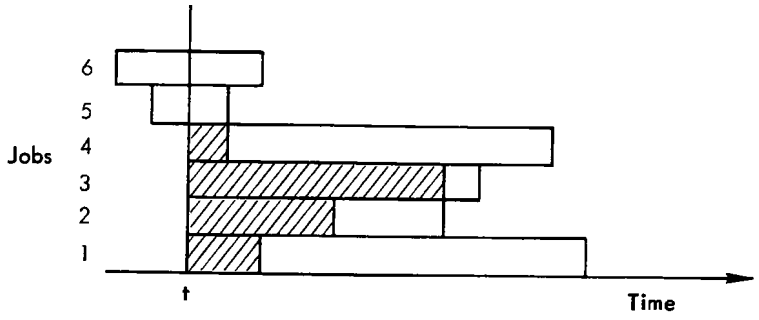


Fig. 3 — Earliest completion time profile for a product

computing priority values? There are many options, but they can generally be broken down to three sets of choices:

(1) Geometric attributes of the diagrams such as lengths of bars, number of bars, or areas.
(2) Choice of jobset to consider: *A*, *UC*, or *US*.
(3) Whether to operate with P's or E's.

Before defining the specific set of rules considered, we state three principles that were confirmed in the study:

First, in queueing systems where minimizing either the average flow time or the average number of jobs in the system is the objective, it is generally true that assignment procedures giving precedence to jobs with small demands for machine time will perform better than those that do not. This can be proved for the single channel queue and other simple examples and has been shown to be true experimentally in more complex situations [Conway and Maxwell, 1962]. As a general strategy, the "shortest processing time" idea should be valid for this assembly shop also, but how to apply it is not clear.

Second, a good assignment procedure cannot consider jobs in isolation; there must be some consideration of properties of the whole product. Otherwise, there often would be large variability in the priority assignment values among jobs within a product. This would result in heavily loaded resources being devoted to jobs that then would wait for long periods in the assembly shop.*

Third, priority values associated with products should be dynamic in that they improve as jobs progress through the shop. This means that completed jobs should be ignored. Since we are always trying to make decisions that will reduce the sum of future completion dates, it makes no sense to penalize a product for having, say, a long job

---

*For a static version of our problem, i.e., a finite number of products, all available at time 0 , with one machine in each machine group, it can be shown that there is an optimal sequence that satisfies a "no passing property," which is equivalent to assigning the same priority value to all jobs within a product. A proof is given in Pritsker et al. [1971].

if that job has been completed. (Note that minimizing average flow time is equivalent to minimizing the sum of completion dates since the arrival dates are fixed.)

Rules Based on Lengths (P or E)

SPT: Shortest Processing Time

$$Z_j(t) = P_j$$

The shortest processing time rule is of interest because it has worked well in other environments, but it violates our second principle.

LJ: Long Job

$$Z_j(t) = \max_{k \in A_{p(j)}(t)} \{P_k\}$$

This rule says to use the processing time of the longest job in a product as the priority value for all its jobs. It violates our third principle.

LJ-US: Longest Unstarted Job

$$Z_j(t) = \max_{k \in US_{p(j)}(t)} \{P_k\}$$

This is a dynamic analog of the previous rule; it assigns to each job in a product a priority value equal to the processing time of the longest unstarted job in the product.

LJSLK-US: Long Job Slack

$$Z_j(t) = \max_{k \in US_{p(j)}(t)} \{P_k\} - P_j$$

This rule sets priority values equal to the differences between the processing time of the longest unstarted job in a product and the processing times of its several members. Slack rules represent attempts to coordinate the progress of jobs within a product by avoiding short jobs when there are longer ones to get underway. Conversely, they give priority to long jobs whose completions are likely to determine the completion date of the product. Unfortunately, simple slack rules such as this one tend to favor the longest jobs

in a great many products without discrimination while the shorter ones are delayed so long that it is they that determine the completion date. Although not investigated in this experiment, it is likely that the slack idea could be incorporated in procedures that do not have this defect.

EPCD-US: Earliest Possible Completion Date Considering Unstarted Jobs

$$Z_j(t) = \max_{k \in US_{p(j)}(t)} \{E_k(t)\}$$

and

EPCD-UC: Earliest Possible Completion Date Considering Uncompleted Jobs

$$Z_j(t) = \max_{k \in UC_{p(j)}(t)} \{E_k(t)\}$$

The first of these is analogous to LJ-US. Taking into account current commitments of resources might result in more realistic appraisals of the completion dates for products. In addition, the UC version might be better than the US version because occasionally a very long job could be on a machine and there would be no hurry about getting other jobs on the same product started. A comparison between LJ-US and EPCD-US is interesting to see whether or not there is any payoff in return for the additional data processing involved in using E's rather than P's. For rules based on lengths, the answer seems to be *yes*.

Rules Based on Job Counts

NUC-SPT: Number of Uncompleted Jobs, with a Shortest Processing Time Tie-Breaker

$$Z_j(t) = 100\, n\{UC_{p(j)}(t)\} + P_j$$

NUS-SPT: Number of Unstarted Jobs, with a Shortest Processing Time Tie-Breaker

$$Z_j(t) = 100\, n\{US_{p(j)}(t)\} + P_j$$

NUS-LJUS: Number of Unstarted Jobs, with a Longest Unstarted Job Tie-Breaker

$$Z_j(t) = 100\, n\{US_{p(j)}(t)\} + \max_{k \in US_{p(j)}(t)} \{P_k\}$$

Since job counts are integers and many ties are to be expected, secondary rules are necessary. The above set are the first rules to consider the vertical dimensions of Figs. 1 and 3 explicitly. They are dynamic in that selecting a job from a product improves the priority status of the other jobs on the same product. They also incorporate the SPT rule in what seems to be a reasonable way. (It turned out that the last of these was the best of the group, but not as good as LJ-US by itself. With the SPT tie-breaker, NUS was slightly better than NUC.)

Rules Based on Areas

Rules in this group attempt to consider the work content of products either by multiplying job counts by job length, or by summing the areas. The latter option seems to work better. Two multiplicative rules are:

NUS×PT: Number of Unstarted Jobs Times Processing Time

$$Z_j(t) = n\{US_{p(j)}(t)\}\, P_j$$

NUS×LJ-US: Number of Unstarted Jobs Times Processing Time of the Longest Unstarted Job

$$Z_j(t) = n\{US_{p(j)}(t)\} \max_{k \in US_{p(j)}(t)} P_k$$

Three additive rules are:

TWC-US: Total Work Content of Unstarted Jobs

$$Z_j(t) = \sum_{k \in US_{p(j)}(t)} P_k$$

TWC-UC: Total Work Content of Uncompleted Jobs

$$Z_j(t) = \sum_{k \in UC_{p(j)}(t)} P_k$$

TWTC-UC: Total Work to Completion for Uncompleted Jobs

$$Z_j(t) = \sum_{k \in UC_{p(j)}(t)} [E_k(t) - t]$$

With the exception of NUS×PT, these five rules performed better than those of the previous groups. They are all highly dynamic, but NUS×PT badly violates our second principle--intra-product variability of priority values becomes tremendous. Rules of this class have similar effects with respect to improving the priority values of a job when another job of the same product is started or completed. We hypothesize that the following may be a significant distinction. In the case of multiplicative rules, the improvement in priority values is frequently independent of the job selected or completed. But with the additive rules, the improvement depends on the size of the job involved in the event. Often, the bigger the job, the more significant the event, and the more is the improvement in priorities.

Modified Multiplicative Rules

Variations in multiplicative rules can be obtained by raising the job count to a power. Two examples are:

NUS(u)×PT: Weighted Product of Number of Unstarted Jobs and Processing Time

$$Z_j(t) = n\{US_{p(j)}(t)\}^u P_j$$

and

NUS(u)LJ-US: Weighted Product of Number of Unstarted Jobs and Length of Longest Unstarted Job

$$Z_j(t) = n\{US_{p(j)}(t)\}^u \max_{k \in US_{p(j)}(t)} \{P_k\}$$

Changing the exponent has the effect of varying the relative weights between the count and time components. For instance, as u increases from zero, NUS(u)×PT goes from SPT toward NUS-SPT. The additional flexibility did not appear valuable, however.

Processing Time by Queue Length

Only one attempt was made to incorporate information on congestion. It was a rule of the form:

TPT×Q(u): Total Processing Time by Queue Length

$$Z_j(t) = \sum_{k \in US_{p(j)}(t)} P_k \, Q_{m(k)}^{u}$$

Compared to "total work content" rules, a positive exponent results in lower relative priority values for products with jobs in large queues. This might be appropriate because such products face more competition, and getting all of their jobs finished quickly would be difficult. With u = 0, this becomes TWC-US, which turned out better than attempts with u = ±0.5, so this particular way of taking queue lengths into consideration seems to lead nowhere. In view of the findings of Maxwell and Mehra [1968], it is likely that useful ways of taking account of congestion could be found.

A Processing-Time-Independent Rule

FASFS: First Arrival At Shop First Served

$$Z_j(t) = R_j$$

This rule is not dynamic and does not use processing time information. But it does represent, in a pure form, a simple attempt at coordination.

Table 1

SUMMARY OF PRIORITY RULES

(Arguments of sets are suppressed)

Symbol	Definition of $Z_j(t)$	Description
SPT	P_j	Shortest processing time.
LJ	$\max_{k\varepsilon A} \{P_k\}$	Product with the shortest long job.
LJ-US	$\max_{k\varepsilon US} \{P_k\}$	Product with the shortest unstarted long job.
LJSLK-US	$\max_{k\varepsilon US} \{P_k\} - P_j$	Smallest difference between longest unstarted job and job in question--long job slack.
EPCD-US	$\max_{k\varepsilon US} \{E_k(t)\}$	Earliest possible completion date among unstarted jobs.
EPCD-UC	$\max_{k\varepsilon UC} \{E_k(t)\}$	Earliest possible completion date among uncompleted jobs.
NUC-SPT	$100\ n\{UC\} + P_j$	Fewest uncompleted jobs with SPT tie-breaker.
NUS-SPT	$100\ n\{US\} + P_j$	Fewest unstarted jobs with SPT tie-breaker.
NUS-LJUS	$100\ n\{US\} + \max_{k\varepsilon US} \{P_k\}$	Fewest unstarted jobs with longest unstarted job tie-breaker.
NUS×PT	$n\{US\}\ P_j$	Smallest product of number of unstarted jobs and processing time.
NUS×LJ-US	$n\{US\} \max_{k\varepsilon US} \{P_k\}$	Product of smallest product of number of unstarted jobs and longest unstarted job.
TWC-US	$\sum_{k\varepsilon US} P_k$	Smallest total work content of unstarted jobs.
TWC-UC	$\sum_{k\varepsilon UC} P_k$	Smallest total work content of uncompleted jobs.
TWTC-UC	$\sum_{k\varepsilon UC} [E_k(t) - t]$	Smallest total work to completion for uncompleted jobs.
NUS(u)×PT	$n\{US\}^u\ P_j$	Smallest weighted product of number of unstarted jobs and processing time.
NUS(u)LJ-US	$n\{US\}^u \max_{k\varepsilon US} \{P_k\}$	Smallest weighted product of number of unstarted jobs and length of longest unstarted job.
TPT×Q(u)	$\sum_{k\varepsilon US} P_k Q^u_{m(k)}$	Smallest sum of weighted product of processing time and queue length.
FASFS	R_j	Earliest arrival at shop.

IV. MEASURES OF PERFORMANCE

The following notation will be used in defining the measures of performance. Let:

N = Number of products in the sample.
i = Index over products.
A_i = Set of jobs in product i .
j = Index over jobs.
$p(j)$ = Product to which job j belongs.
R_i = Release (arrival) date of product i .
D_i = Due date of product i . $D_i = R_i + 0.667$.
c_j = Completion date of job j .

Using these definitions, some derived quantities are:

C_i = Completion date of product i . $C_i = \max_{j \in A_i} \{c_j\}$.
F_i = Flow time of product i . $F_i = C_i - R_i$.
T_i = Tardiness for product i . $T_i = \max \{0, C_i - D_i\}$.
s_j = Job shop time for job j . $s_j = c_j - R_{p(j)}$.
a_j = Assembly shop time for job j . $a_j = C_{p(j)} - c_j$.

The product completion date, C_i, is the date at which the last job on the product is finished, and flow time, F_i, is the time the product spends in the shop. The measures: *flow time mean* and *flow time variance* are simply the average and the variance around that average of the F_i's. *Tardiness* is the number of days late for a product completed after its due date and is zero if the product is completed before its due date. The *tardiness mean* is the average of the T_i's over late products. That is, it is a conditional mean; early jobs are not counted in the denominator. *Percent tardy* gives the proportion of products that were finished beyond their due dates. The *job shop mean* and *assembly shop mean* are defined as

$$\bar{s} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n\{A_i\}} \sum_{j \epsilon A_i} s_j \quad \text{and} \quad \bar{a} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n\{A_i\}} \sum_{j \epsilon A_i} a_j \ .$$

Notice that in these definitions, individual jobs are not weighted equally; jobs in small products count more than those in large ones. These definitions were adopted because $\bar{F} = \bar{s} + \bar{a}$. This way of decomposing $\bar{F}$ is useful because it allows comparisons on the basis of two primary effects: how well a procedure succeeds in getting jobs through the job shop and how well it does in coordinating the completion of jobs within a product.

Computer time is reported in order to give an indication of how the rules vary in computational requirements.

V. RESULTS

Results from the simulation trials are given in Table 2. This section contains some tentative conclusions derived from examining the data. Conclusions are tentative because in many cases we are looking at small differences that are not statistically significant (see the following section).

NEED FOR COORDINATION

The SPT rule, which has proved to be the best rule in the case of independent jobs, was the worst of those tried. It was beaten even by FASFS, which neither uses processing time information nor is dynamic. About all it does do is give a common priority value to all jobs within a product. The superiority is due to reduced assembly shop time, even though the job shop time is worse.

Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
SPT	.843	.199	.644
FASFS	.788	.312	.476

These were the first two runs made, and they showed immediately that the presence of assembly constraints does introduce a problem. A modest attempt at coordination combined with random selection with respect to processing times outweighs spectacular performance in the job shop when there is no coordination.

STATIC VS. DYNAMIC RULES

A direct comparison between static and dynamic rules is available from only two trials:

	Static				Dynamic		
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$	Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
LJ	.861	.396	.464	LJ-US	.676	.369	.308

Table 2

EXPERIMENTAL RESULTS

Rule	Flow Time Mean	Job Shop Time Mean	Assembly Shop Time Mean	Flow Time Variance	Percent Tardy	Tardiness Mean	Computer[a] Time (min)
Twenty Machine Groups							
SPT	.8428	.1991	.6437	.6487	39.87	.7138	32:39
NUC-SPT	.6883	.2713	.4170	.1763	35.43	.4093	31:54
LJSLK-US	.8180	.3544	.4636	.1763	53.49	.4010	78:40
FASFS	.7880	.3125	.4755	.1461	50.89	.3629	34:00
Eight Machine Groups							
SPT	.8610	.2783	.5827	1.7504	35.78	1.1402	1:30
LJ	.8607	.3962	.4645	3.0686	25.70	1.6730	1:43
LJ-US	.6765	.3688	.3077	1.4237	23.17	1.2054	1:33
EPCD-US	.6515	.3731	.2784	1.3425	22.33	1.1648	3:34
EPCD-UC	.6433	.3732	.2701	1.3652	21.89	1.1642	3:37
NUC-SPT	.7593	.4145	.3448	.9647	35.78	.8320	1:30
NUS-SPT	.7408	.4154	.3254	1.0609	31.67	.9435	1:33
NUS-LJUS	.7315	.4071	.3244	.8356	33.11	.8452	1:32
NUS×PT	.7066	.2935	.4131	1.0049	28.56	.9691	1:33
NUS(2)×PT	.6906	.3377	.3529	.7929	29.78	.8601	2:01
NUS×LJ-US	.6437	.3806	.2631	.9495	25.06	.9380	2:16
NUS(.9)×LJ-US	.6450	.3790	.2660	.9772	25.00	.9621	2:17
NUS(1.2)×LJ-US	.6537	.3856	.2681	.8127	25.56	.9516	2:16
NUS(1.5)×LJ-US	.6637	.3937	.2700	.8247	26.22	.9490	2:17
TWC-US	.6220	.3601	.2619	.9125	23.72	.9324	1:28
TWC-UC	.6322	.3696	.2626	.7811	25.11	.8819	1:56
TWTC-UC	.6436	.3814	.2622	1.1304	25.06	.9495	4:05
TPT×Q(.5)	.6279	.3650	.2629	.8520	23.61	.9580	5:48
TPT×Q(-.5)	.6430	.3648	.2782	.8671	25.11	.9340	5:45

[a]Early runs, those with the 20 machine groups, were done on an IBM 7044; the later series was carried out on an IBM 360/65. Also, about 7½ times as many jobs were processed in the twenty-group trials as were in the eight-group trials.

The difference is so great that no other trials were made to illustrate the point. Notice that only SPT exceeded LJ in assembly shop time. Two static variants of LJSLK were tried in the earlier runs with twenty machine groups, but congestion was so great that the computer ran out of memory space.

JOB VS. PRODUCT ATTRIBUTES

Processing time is the only meaningful job attribute and processing time of the longest unstarted job is a closely related product attribute. In every case where these two attributes were used in a similar way, the long job version was superior.

	Job		
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
SPT	.861	.278	.583
NUS×PT	.707	.294	.413
NUS-SPT	.741	.415	.326

	Product		
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
LJ-US	.676	.369	.308
NUS×LJUS	.644	.381	.263
NUS-LJUS	.732	.407	.324

As expected, LJ-US makes its gains in the assembly shop when compared to SPT. The same holds true for the multiplicative rules. But that NUS-LJUS seems at least as good as NUS-SPT in the job shop and no better in the assembly shop is a surprise. Notice that NUS×LJUS did nearly as well as the TWC rules with respect to assembly shop time, probably because the former is an approximation of the latter.

ROLE OF STARTED-BUT-NOT-COMPLETED JOBS

Whether data on jobs in S should be included in priority values appears to depend on the type of rule. Length-type rules, which are attempts to create self-fulfilling prophecies in that they try to estimate possible completion times, work better if jobs in S are included. For example, compare the two versions of EPCD:

	Include S		
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
EPCD-UC	.643	.373	.270

	Exclude S		
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
EPCD-US	.652	.373	.278

On the other hand, job count and work content rules work better if jobs in S are ignored:

Include S			
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
NUC-SPT	.759	.414	.345
TWC-UC	.632	.370	.263

Exclude S			
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
NUS-SPT	.741	.415	.325
TWC-US	.622	.360	.262

Here it seems that the act of selecting a job is a significant event to a product, and the rules are more sensitive if priorities are adjusted at job starts rather than completions. For the job count rules the advantage was in the assembly shop, and for the work content rules, the advantage was in the job shop.

PROCESSING TIMES VS. COMPLETION TIMES

The choice of whether to use processing times (P's) or projected completion dates (E's) also depends on the class of the rule. Length rules work better with completion dates and work content rules are better when processing times are used. For work content rules, the additional information in completion dates is probably just noise. The comparisons may be seen from the following:

Processing Times			
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
LJ-US	.676	.369	.308
TWC-UC	.632	.370	.263

Completion Dates			
Rule	$\bar{F}$	$\bar{s}$	$\bar{a}$
EPCD-US	.652	.373	.278
TWTC-UC	.644	.381	.262

THE BEST RULE

Of the rules tested, TWC-US resulted in the smallest average product flow time. One may hypothesize that in addition to using processing time information, being dynamic, and being based on a product attribute, it is also "properly" sensitive to key events. That is, priorities of all remaining unstarted jobs in a product get a larger boost when a long job is started than when a smaller one is.

VI. STATISTICAL SIGNIFICANCE

Assessing the statistical significance of results in simulation experiments such as this one is made difficult by the fact that the flow times of individual products are an autocorrelated sequence. Thus, dividing the sample variance of flow times by the sample size would produce a gross underestimate of the variance of sample means. To illustrate, a trial of NUC-SPT under conditions similar to those of the early trials was made with a sample size of 1800 products. The sample variance of flow times was 0.29, which, divided by 1800, gives 0.00016. Then an estimate of the variance of sample means was obtained by computing the power spectrum of the sequence [Fishman and Kiviat, 1967]. The value obtained was 0.0038, more than twenty times the estimate under the assumption of independence.

A simple way of achieving more realistic estimates, discussed by Conway [1963], was carried out in this experiment. The method is to partition the products into groups made up of consecutively arriving products. The larger the groups, the less are the group averages correlated, so that with sufficiently large groups, the variability of the grand mean can be estimated from the sample variance of the group averages. On the other hand, the groups should be sufficiently numerous that not too many degrees of freedom are lost. Estimates of standard deviations of average flow times for seven rules are given in the second column of figures in Table 3. These were based on 18 groups of 100 products each.

There really is little interest, however, in setting confidence intervals for the flow time means. We are much more interested in assessing the significance of differences. Because all rules were run against the same set of products, the sequences of group averages are highly correlated. This works very much in our favor because it reduces the variance of differences. For the seven rules represented in Table 3, Pearson product-moment correlation coefficients among the 21 pairwise sequences of group averages are given above the diagonal. Below the diagonal are given t statistics with 17 degrees of freedom computed from each of the 21 sequences of differences. These serve to suggest which differences are significant.

Table 3

CORRELATIONS (NE) AND t STATISTICS (SW) COMPARING GROUPED FLOW TIMES IN GROUPS OF 100

Rule	Flow Time Mean	Std. Dev. of Grand Mean	Rule						
			TWC-US	EPCD-UC	NUS×LJ-US	EPCD-US	LJ-US	NUS-SPT	NUC-SPT
TWC-US	.622	.0328		.904	.895	.914	.898	.903	.916
EPCD-UC	.643	.0395	1.190		.841	.990	.987	.903	.902
NUS×LJ-US	.644	.0374	1.241	.016		.832	.841	.930	.909
EPCD-US	.652	.0424	1.559	1.224	.322		.991	.908	.907
LJ-US	.676	.0451	2.471	3.629	1.305	3.746		.909	.906
NUS-SPT	.741	.0508	4.533	4.174	4.888	4.021	2.957		.991
NUC-SPT	.759	.0492	5.753	5.200	5.202	5.051	3.907	2.599	

VII. RELATIONSHIP TO AIR FORCE MAINTENANCE

Although this report uses generic terminology customarily employed in the literature of job shop scheduling, the research was motivated by Air Force maintenance problems. Our *shop* represents the base maintenance complex and the *machine groups* represent various shops and work centers. Individual *machines* are specialists belonging to a work center. A *product* is an airplane and the *jobs* represent malfunctions or discrepancies to be cleared.

Our primary interest is in controlling unscheduled maintenance on large aircraft. Most of this type of maintenance results from flying; so a *product arrival* simulates a crew debriefing. A large airplane can be expected to have several malfunctions, and to a considerable extent, they can be worked on independently. To begin processing a job is, in the real world, to dispatch a team of specialists to the aircraft.

Our performance criterion, *minimizing average product flow time*, corresponds to maximizing the average number of operationally ready aircraft. Much theoretical work in scheduling is oriented toward meeting arbitrary due dates, which would correspond to getting each airplane ready in time for its next mission. We believe this is a less desirable orientation because committing specific aircraft numbers to missions far in advance of takeoff times steals flexibility and the schedules actually produced tend to be poor with respect to the operational readiness criterion. (This need not be the case, but it is typical.)

Anyone familiar with aircraft maintenance would recognize this model as a tremendous oversimplification. To mention just a few complications: actual repair times, and even the true nature of malfunctions when first reported, are not known in advance. Malfunctions can be discovered in the course of fixing others or at any time. Work centers are staffed by more than two specialists, they have varying kinds and degrees of skill, and are often dispatched in teams. Sometimes work centers must cooperate. Jobs sometimes cannot be done

concurrently because of space restrictions or safety requirements. Our model considers just one kind of resource, manpower, but maintenance control is also concerned with parts, aerospace ground equipment, and facilities.

Given that the real world of base maintenance is a very complex place, we must face two questions: "Should we not be working with a more realistic model?" and "Have we learned anything?"

More realistic models do exist; four of them: BOMS [Ginsberg and King, 1964], SAMSOM [Smith, 1964], PLANET [Voosen, 1965], and L-COM [Fisher et al, 1968], were developed at Rand. (Indeed, our simple model has always gone by the informal name of "Little BOMS.") But these highly complex models are just not reasonable vehicles for providing insights and exploring first-order effects. They have large appetites for data and computational resources, and so much is going on in them that it is difficult to analyze results. On the other hand, we believe that experimentation with the grossly simplified model has provided insights that could be of considerable value in dealing with at least a portion of the real problem.

Every scheduling environment is unique and there are few universal truths. The best that a scheduler or system designer can do is attempt to synthesize a solution out of methods and qualitative notions that have emerged from studies such as this one. (This is why we make no apologies for not scaling our model to the real world--that would just change the size of the numbers without changing the conclusions, and the results would still not be accurate quantitative predictors.) We hope that Little BOMS will have added one more building block that may prove useful.

REFERENCES

Conway, R. W., "Some Tactical Problems in Digital Simulation," *Management Science*, Vol. 10, No. I, September 1963.

Conway, R. W., *An Experimental Investigation of Priority Assignments in a Job Shop*, The Rand Corporation, RM-3789, February 1964.

Conway, R. W., "Priority Dispatching and Work-in-Process Inventory in a Job Shop," *J. Ind. Eng.*, Vol. 16, No. 2, March 1965.

Conway, R. W., and W. L. Maxwell, "Network Dispatching by the Shortest Operation Rule," *Operations Research*, Vol. 10, No. 1, February 1962.

Conway, R. W., W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison Wesley, Reading, Mass., 1967.

Fisher, R. R., W. W. Drake, J. J. Delfausse, A. J. Clark, and A. L. Buchanan, *The Logistics Composite Model: An Overall View*, The Rand Corporation, RM-5544-PR, May 1968.

Fishman, G. S., and P. J. Kiviat, "The Analysis of Simulation Generated Time Series," *Management Science*, Vol. 13, No. 7, March 1967.

Ginsberg, A. S., and B. A. King, *Base Operations-Maintenance Simulator*, The Rand Corporation, RM-4072-PR, August 1964.

Markowitz, H. M., B. Hausner, and H. W. Karr, *SIMSCRIPT: A Simulation Programming Language*, The Rand Corporation, RM-3310-PR, November 1962.

Maxwell, W. L., *Priority Dispatching and Assembly Operations in a Job Shop*, The Rand Corporation, RM-5370-PR, October 1969.

Maxwell, W. L., and M. Mehra, "Multiple-Factor Rules for Sequencing with Assembly Constraints," *Naval Research Logistics Quarterly*, Vol. 15, No. 2, June 1968.

Pritsker, A. A. B., L. W. Miller, and R. J. Zinkl, "Sequencing n Products Involving m Independent Jobs on m Machines," *AIIE Transactions*, Vol. 3, No. 1, March 1971.

Siegel, Gerald B., "An Investigation of Job Shop Scheduling for Jobs with Assembly Constraints," Ph.D. Dissertation, Cornell University, Ithaca, New York, December 1971.

Smith, T. C., *SAMSON: Support-Availability Multi-System Operations Model*, The Rand Corporation, RM-4077-PR, June 1964.

Voosen, B. J., *Simulation and Evaluation of Logistics Systems*, The Rand Corporation, RM-4589-PR, September 1965.