# SCHEDULING WITH PARALLEL PROCESSORS AND LINEAR DELAY COSTS

Kenneth R. Baker
*North Carolina State University*

and

Alan G. Merten
*the University of Michigan*

**ABSTRACT**

This paper deals with the sequencing problem of minimizing linear delay costs with parallel identical processors. The theoretical properties of this m-machine problem are explored, and the problem of determining an optimum scheduling procedure is examined. Properties of the optimum schedule are given as well as the corresponding reductions in the number of schedules that must be evaluated in the search for an optimum. An experimental comparison of scheduling rules is reported; this indicates that although a class of effective heuristics can be identified, their relative behavior is difficult to characterize.

## 1. INTRODUCTION

Two basic facets of scheduling are the allocation of resources and the sequencing of tasks. In much of the development of scheduling methodology, it has been helpful to simplify the resource structure in order to focus on problems of sequence. Thus in the case of single-machine finite sequencing and in the case of job-shop sequencing, it is usually assumed that each requirement for processing involves a specified, unique resource (Conway, Maxwell, Miller [5]). A first step in treating resource flexibility then is to deal with one resource type and parallel capability. This study deals with the problem of minimizing mean weighted flowtime with parallel machines and with independent nonpreemptable tasks.

The model to be considered involves $m$ machines and $n$ jobs. The machines are identical and each is capable of processing at most one job at a time. The $n$ jobs are independent (that is, no precedence relations exist among them), simultaneously available at time zero, and can each be processed by at most one machine at a time. In addition, job $j$ has associated with it a processing time (denoted $p_j$), known in advance, and a weighting factor ($w_j$), reflecting its value or importance.

For job $j$, the flowtime ($F_j$) denotes the time spent in the system until completion. The performance measure of interest is mean-weighted flowtime:

$$\bar{F}_w = \frac{\sum_{j=1}^{n} w_j F_j}{\sum_{j=1}^{n} w_j}.$$

Another way of looking at the same problem is to suppose that each job has an associated delay cost per unit time spent in the system (McNaughton [15]). If $w_j$ denotes this unit delay cost, then under

a given schedule the total delay cost accumulated for job $j$ is $w_jF_j$. The problem of minimizing total delay cost for the set of $n$ jobs is identical to the problem of minimizing $\bar{F}_w$.

For $m = 1$, the well-known result is that $\bar{F}_w$ is minimized by processing the jobs in nondecreasing order of the ratio $p_j/w_j$. This ordering will be referred to as weighted shortest processing time (WSPT) sequencing. The sequence which maximizes $\bar{F}_w$ is antithetical sequence, weighted longest processing time (WLPT). For $m > 1$, comparable results do not exist, largely due to the presence of a resource allocation problem superimposed upon the sequencing problem.

Models involving $\bar{F}_w$ as a performance measure have been employed in a diverse set of applications, as represented by several of the references (Riesel [22], Merten [17], Bowdon [3], Coffman and Muntz [4], Baker [2], and Grieshop [10]). Some of these sources treat models which explicitly contain parallel processors. In those cases where the discussion is limited to single processor models, it is not difficult to recognize that the parallel processor case is an important and realistic extension of the specialized model. Therefore, the scheduling problem treated in this paper has a very broad spectrum of potential application areas.

## 2. THEORETICAL RESULTS FOR THE $m$-MACHINE PROBLEM

The search for the sequence that minimizes mean flowtime and mean waiting time theoretically must consider all possible schedules of the $n$ jobs on the $m$ machines. This search would be based on the relative weights and processing times of the jobs just as it is in the one-machine case. It is possible, however, to reduce the number of schedules that must be investigated because of certain dominance properties, properties of the mean as a performance function, and symmetries. A schedule of the $n$ jobs on the $m$ machines can be viewed as taking place in the following phases:

(1) partition the $n$ jobs into $m$ sets (machines)

(2) order the jobs on each of the $m$ machines

Because of the linear properties of the mean, the mean flowtime for the $n$ jobs on the $m$ machines is the sum of the mean times for each of the $m$ machines. Therefore, to minimize the mean flowtime over all machines, the mean flowtime must be minimized on each machine (Eastman, Even, and Isaacs [7]). Given that a subset of the jobs is assigned to a particular machine, the optimum sequence for this machine corresponds to the WSPT ordering. Therefore, the number of schedules that must be investigated is no greater than the number of ways of assigning $n$ jobs to $m$ machines since, from the one-machine results, WSPT is known to be optimal on a single machine.

The next reduction in the size of the set of schedules to be investigated comes as a result of the observation that an optimum schedule cannot contain an empty machine. For if a schedule were to include an empty machine, a schedule which has a lower mean flowtime can be obtained by taking a job from a machine where there is more than one job and moving it to the previously empty machine (the waiting-time for the job that is moved is reduced from a positive number to zero). The process of eliminating empty machines can be continued until there are no empty ones left.

Finally, since the machines are identical prior to the assignment of the jobs, certain schedules can be ignored since they are indistinguishable from other schedules.

In order to describe sets of schedules, let

$Z$ = set of all possible ways of scheduling $n$ jobs on $m$ machines;

$Z_p$ = set of all possible ways of scheduling $n$ jobs on $m$ machines if the sequence within a machine is ignored (use the WSPT sequence within each machine);

$Z_e$ = set of all possible ways of scheduling $n$ jobs on $m$ machines using WSPT for each machine and excluding the cases where there is one or more unused machines;

$Z_m$ = set of all possible ways of scheduling $n$ jobs on $m$ machines using WSPT, excluding the unused machine cases and ignoring the indistinguishable schedules.

Therefore, $Z \supset Z_p \supset Z_e \supset Z_m$. The problem is now to find $N(A)$, the number of elements in each of these sets $A$.

Table 1 shows the expressions for the sizes of the sets defined above and gives examples for some small values of $n$ and $m$. The derivation of these expressions is given in Merten [17]. Some of these results were derived from previous work in combinatorial analysis (Feller [8], Knuth [14], and Abramovitz and Stegun [1]). Even for these small numbers of jobs and machines, it is clearly important to isolate the subset of schedules that contains the sequence that minimizes mean waiting-time.

The following additional results have been shown for the general $m$-machine problem:

1. It is sufficient to consider schedules in which there is no preemption of jobs (McNaughton [15]).

2. A lower bound $B(m)$, on the optimum solution can be obtained as follows (Eastman, Even, Issacs [7]): Let $B(1)$ denote optimum value of $\bar{F}_w$ for the given job set when $m = 1$ (given by WSPT). Let $B(n)$ denote the optimum value of $\bar{F}_w$ when $m = n$ (given by assigning each job to a different machine, so that $F_j = p_j$). Then

$$B(m) = \max\left\{B(n), \frac{1}{m}B(1) + \frac{m-1}{2m}B(n)\right\}.$$

3. The problem can be formulated as a dynamic programming problem (Held and Karp [11]). This formulation can include the case where the job execution time may differ depending on which machine is used. While the dynamic programming formulation does lead to some reduction in the computation required to find the optimum solution, the procedure is still inadequate for solving large problems.

TABLE 1. *Number of Ways of Sequencing* n *Jobs on* m *Machines as a Function of the Schedule Set*

| | $N(Z)$ | $N(Z_p)$ | $N(Z_e)$ | $N(Z_m)$ |
|---|---|---|---|---|
| $(n, m)$ | $n!\binom{n+m-1}{m-1}$ | $m^n$ | $\left\{{n \atop m}\right\} m!$ | $\left\{{n \atop m}\right\}$ |
| (3, 2) | 24 | 8 | 6 | 3 |
| (4, 2) | 120 | 16 | 14 | 7 |
| (3, 3) | 60 | 27 | 6 | 1 |
| (4, 3) | 360 | 81 | 36 | 6 |

When all weights are equal, the optimal schedule for the $m$-machine problem can be constructed by arranging the jobs in nondecreasing order of processing time and then assigning the jobs in this order to a machine as soon as one is made available. In practice, this would correspond to the creation of a

single job-queue in which shortest-first priority prevails. Whenever a machine became available, it would be assigned the highest priority job among those remaining in the queue.

Several alternate constructions will yield optimum solutions as well, and it is useful to consider a different viewpoint. As discussed by Conway, Maxwell, and Miller [5, pp. 77–78] an optimal schedule for the equal-weighting problem can be found as follows:

1. Find the jobs with the $m$ longest processing times and assign them in any order to $m$ different machines.
2. Remove the assigned jobs from consideration and repeat step 1 until all jobs are assigned.
3. At each machine, process the jobs in shortest-first sequence.

Similarly, it has been shown, when all the job processing-times are equal, the optimal allocation has the property that the $m$ jobs with the largest weights are in the first positions on the $m$ machines, the jobs with the $m$ largest weights of those remaining are in the second positions on the $m$ machines, and so on until all the jobs have been assigned a position. (Merten [17]).

## 3. HEURISTIC SCHEDULING PROCEDURES

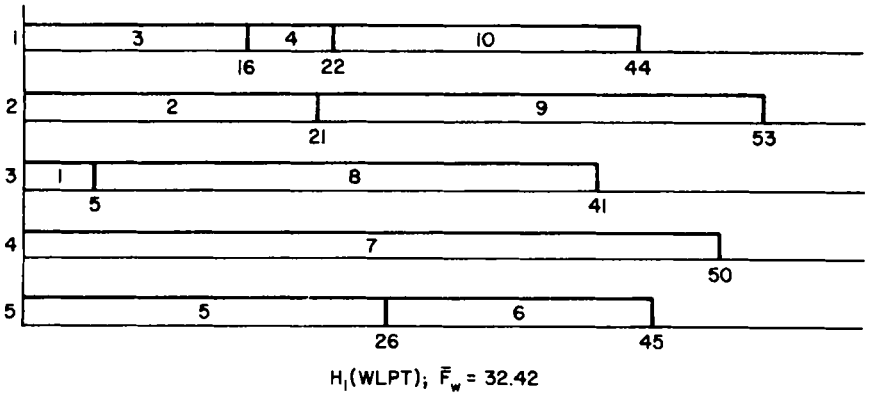
The foregoing discussion of the equal-weighting problem serves to identify two basic approaches to the more general problem: a *one-at-a-time* job assignment strategy and an *m-at-a-time* job assignment strategy.

Under the one-at-a-time strategy, which is called heuristic $H_1$, a priority rule is selected in order to form a ranked list of the jobs. The machine with the smallest amount of scheduled processing is then assigned the first job on the list. This step is repeated until all jobs are assigned to machines and then the jobs assigned to each machine are ordered by WSPT. To illustrate how heuristic $H_1$ works, consider the 10-job set shown in Table 2 and suppose that five machines are available. Also, suppose that the priority rule WLPT is selected to form the initial ranked list (so that the jobs are considered in the order 10, 9, 8, . . ., 2, 1.) At the start, no processing has been assigned to any machine, so the first five jobs on the list are assigned to five different machines (see Table 3). At this stage, the vector of total processing commitments assigned to each machine is (22, 32, 41, 50, 19). Since the minimum occurs for machine 5, the next job (job 5) is assigned to machine 5. The updated vector of machine commitments is (22, 32, 41, 50, 45). Now the minimum occurs for machine 1, and so the next job (job 4) is assigned to machine 1. The details of the procedure are presented in Table 3, and the final sequence generated by this combination of $H_1$ and WLPT is shown in Figure 1. If WLPT were replaced by some other priority rules for ranking the jobs initially, $H_1$ might lead to a different schedule, with a different value of $\bar{F}_w$.

TABLE 2. *A 10-Job Data Set, in which the Jobs Are Numbered in WSPT Order*

| Job | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_j$ | 5 | 21 | 16 | 6 | 26 | 19 | 50 | 41 | 32 | 22 |
| $w_j$ | 4 | 5 | 3 | 1 | 4 | 2 | 5 | 4 | 3 | 2 |
| $p_j/w_j$ | 1.25 | 4.2 | 5.3 | 6 | 6.5 | 9.5 | 10 | 10.2 | 10.7 | 11 |

Under the basic $m$-at-a-time strategy, which is called heuristic $H_m$, a priority rule is again selected to form a ranked list of the jobs. The first $m$ jobs on the list are assigned to $m$ different machines. The




$H_1$(WLPT); $\bar{F}_w = 32.42$

FIGURE 1.

TABLE 3

1. Initial job list {10, 9, 8, 7, 6, 5, 4, 3, 2, 1 }

2. Assignment phase.

| *Processing commitments* | *Job* | *Machine assinged* |
|---|---|---|
| (0, 0, 0, 0, 0) | 10 | 1 |
| (22, 0, 0, 0, 0) | 9 | 2 |
| (22, 32, 0, 0, 0) | 8 | 3 |
| (22, 32, 41, 0, 0) | 7 | 4 |
| (22, 32, 41, 50, 0) | 6 | 5 |
| (22, 32, 41, 50, 19) | 5 | 5 |
| (22, 32, 41, 50, 45) | 4 | 1 |
| (28, 32, 41, 50, 45) | 3 | 1 |
| (44, 32, 41, 50, 45) | 2 | 2 |
| (44, 53, 41, 50, 45) | 1 | 3 |

3. Reorder at each machine by WSPT

| *Machine* | *Sequence* |
|---|---|
| 1 | 3–4–10 |
| 2 | 2–9 |
| 3 | 1–8 |
| 4 | 7 |
| 5 | 5–6 |

next $m$ jobs on the list are assigned to $m$ unique machines and so on, until all jobs are assigned. Then WSPT sequencing is applied to each machine. The $m$-way assignment required at each stage can be specified in more detail. Consider the situation in which the assignment step has been repeated $s$ times, so that $s \cdot m$ jobs are assigned. Taking these assignments to be fixed, consider the subproblem in which it is desired to allocate the next $m$ jobs at stage $s+1$ so that mean weighted flowtime is minimized for all assigned jobs. It is not difficult to show that the optimum allocation in this subproblem is the assignment of the job with the largest weighting factor to the machine with the next smallest processing commitment, and so on. This assignment mechanism is incorporated in $H_m$. To illustrate this heuristic, consider the example introduced above, and again let WLPT be used to rank the jobs initially. At the first stage, jobs 6–10 are assigned to different machines (see Table 4); this yields a machine commit-

TABLE 4.

1. Initial job list {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}

2. Assignment phase

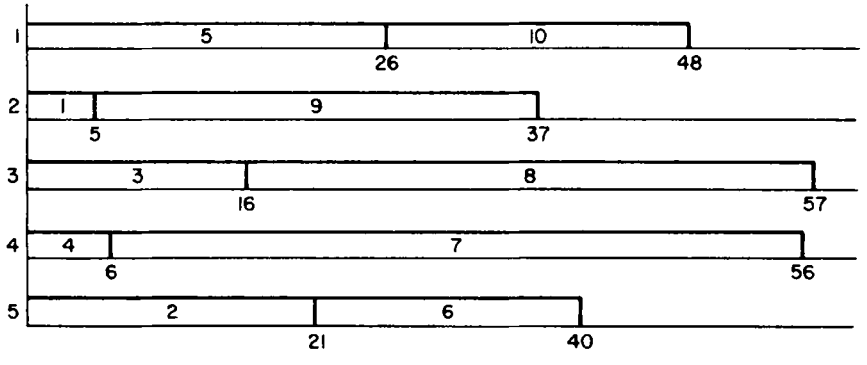
| *Stage* | *Processing commitments* | *Jobs*($w_j$) | *Machine assigned* |
|---|---|---|---|
| 1 | (0, 0, 0, 0, 0) | 10 (2) | 1 |
| | | 9 (3) | 2 |
| | | 8 (4) | 3 |
| | | 7 (5) | 4 |
| | | 6 (2) | 5 |
| 2 | (22, 32, 41, 50, 19) | 5 (4) | 1 |
| | | 4 (1) | 4 |
| | | 3 (3) | 3 |
| | | 2 (5) | 5 |
| | | 1 (4) | 2 |

3. Reorder at each machine by WSPT

| *Machine* | *Sequence* |
|---|---|
| 1 | 5–10 |
| 2 | 1–9 |
| 3 | 3–8 |
| 4 | 4–7 |
| 5 | 2–6 |

ments vector of (22, 32, 41, 50, 19). At the second stage the machines are ordered smallest-first by this commitment (5–1–2–3–4) and the jobs are ordered largest-first by weighting factor, ties being broken arbitrarily (2–5–1–3–4). This leads to the assignment of job 2 to machine 5, job 5 to machine 1, job 1 to machine 2, job 3 to machine 3, and job 4 to machine 4. The details of this heuristic are given in Table 4 and Figure 2. Once again, the use of a priority rule different from WLPT might lead to a different schedule. In any case, when $n$ is an even multiple of $m$ (as in the example problem), heuristic $H_m$ will always assign the same number of jobs to every machine.

One variation of this form of $H_m$ is to relax the restriction that the $m$ jobs must be assigned to different machines at each stage. In this case, a possible heuristic is to treat the $m$ jobs in decreasing order of their weighting factors and to assign them one at a time to the machine with the smallest processing




$H_m$(WLPT); $\bar{F}_w$ = 32.67

FIGURE 2.

commitment. Under this heuristic, called $H_x$, it is possible that several of the $m$ jobs considered at a given stage will be assigned to the same machine. We were surprised, however, to find in our experimentation that $H_x$ was consistently less effective than the other two heuristics.

The priority rule at the heart of each heuristic can be selected from a variety of orderings which are potentially effective in sequencing. At least five basic priority rules are of interest: shortest processing time (SPT), longest processing time (LPT), their weighted versions (WSPT and WLPT), and largest weighting factor ($W$). Used in conjunction with the three heuristics $H_1$, $H_m$, and $H_x$, they yield 15 distinct scheduling procedures with various combinations being denoted as $H_1$ (WLPT), $H_m$ ($W$), etc. The remainder of the discussion deals with the solutions generated by these procedures.

For the 10-job example of Table 2, an examination of the 15 scheduling procedures reveals that the best schedule is not one of those in Figures 1–3, but is one produced by $H_m$ (WSPT), with $\bar{F}_w = 32.30$. (In order to determine an optimum schedule, it would be necessary to examine $N(Z_m) = 42{,}525$ schedules in this problem.) The second best schedule is produced by $H_1$ (WLPT) and the third best by $H_1$ (LPT). It is interesting to observe, however, that for the same job set with a different number of machines, the relative ordering of the scheduling procedures is somewhat different. Table 5 displays results for $2 \leq m \leq 6$. Two important properties are evident: first, several different procedures produce best schedules at least once in the five different problems; and second, the set of three best procedures is different for every value of $m$. There is no clear-cut choice for the best scheduling rule, nor is there yet even a convincing choice between heuristics $H_1$ and $H_m$.

This instability of relative performance among scheduling procedures might well be particularly characteristic of small problems. When $n$ is small, a change in the scheduling of one or two jobs can represent a significant change in the overall performance measure, whereas this is much less likely to be the case when $n$ is large. As illustrated by this example problem, when the job set is small there may be considerable nonuniformity in the effectiveness of a particular rule. It is doubtful that a truly optimum procedure will exist among the heuristic procedures examined here.

To shed some light on the question of the effect of problem size, a more detailed investigation of larger problems was carried out.

TABLE 5. *Rankings of the Three Best Rules for Different Values of* m.

| $m$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| Best | $H_1$ (WSPT)[a] | $H_1$ (LPT) | $H_1$ (WSPT) | $H_m$ (WSPT) | $H_m$ (W)[a] |
| Second | $H_m$ (WSPT)[a] | $H_m$ (WSPT) | $H_1$ (WLPT) | $H_1$ (WLPT) | $H_1$ (W)[a] |
| Third | $H_x$ (WSPT) | $H_1$ (SPT) | $H_m$ (W) | $H_1$ (LPT) | $H_1$ (WSPT)[a] |

[a] Ties.

## 4. EXPERIMENTS WITH LARGE JOB SETS

Six large job sets, of size $n = 100$ jobs, were constructed for further experimentation. The job sets were generated as follows: (a) the processing times in each set consisted of random samples drawn from a distribution with a mean of 50, (b) the weights in each set were independent of the processing

times and were samples drawn from a distribution with a mean of five. The job sets were distinguished by the forms of the distribution, as shown in Table 6. For each job set, the same jobs were scheduled for $m$ parallel machines, where $m$ was again varied from 2 to 6. With six data sets and five versions of parallelism, 30 different problems were posed, and each of the 15 scheduling procedures was tested on each of the 30 problems.

Detailed results for data set four are displayed in Table 7. For each combination of scheduling procedure and number of machines, the schedule value is given as the difference between $\bar{F}_w$ and the lower bound, along with its ranking among the 15 procedures. This particular data set is perhaps a typical among the six tested, but it does serve to highlight many of the characteristics of the problem. Specifically, the results show that there is only a limited amount of dependence in the rankings as $m$ is varied. Although the five different values of $m$ do not generate completely independent sets of observations, they do convey much more information than the results for any single value of $m$ alone.

Three different rules emerged as best in this job set: $H_m$ (WSPT), $H_1$ (WLPT), and $H_1$ (WSPT). Thus two different heuristics $H_1$ and $H_m$ emerged as best. This configuration dramatically illustrates

TABLE 6

| Data Set | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $p$-distribution | $U(0, 100)$ | $N(50, 10)$ | $U(0, 100)$ | $N(50, 10)$ | $U(0, 100)$ | $X(50)$ |
| $w$-distribution | $U(0, 10)$ | $U(0, 10)$ | $N(5, 1)$ | $N(5, 1)$ | $X(5)$ | $X(5)$ |

Notation: $U(a, b)$ Uniform on the interval $a$ to $b$.
$N(a, b)$ Normal with mean $a$ and standard deviation $b$.
$X(a)$ Exponential with mean $a$.

TABLE 7

| $m$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| $H_1$ (SPT) | 0.65 8 | 1.23 9 | 1.91 9 | 2.39 9 | 1.55 5 |
| (WSPT) | 0.10 2 | 0.18 2 | 0.18 1 | 0.31 2 | 0.29 1 |
| (LPT) | 2.04 11 | 1.32 10 | 1.55 8 | 1.90 7 | 2.43 11 |
| (WLPT) | 0.12 4 | 0.16 1 | 0.19 2 | 0.35 4 | 0.33 2 |
| ($W$) | 0.51 7 | 3.02 13 | 2.18 11 | 2.17 8 | 1.94 8 |
| $H_m$ (SPT) | 20.89 14 | 15.91 14 | 13.91 14 | 12.07 15 | 11.98 14 |
| (WSPT) | 0.09 1 | 0.22 3 | 0.22 3 | 0.27 1 | 0.38 3 |
| (LPT) | 21.53 15 | 16.40 15 | 14.62 15 | 11.18 14 | 12.80 15 |
| (WLPT) | 0.11 3 | 0.23 4 | 0.27 4 | 0.34 3 | 0.17 4 |
| ($W$) | 2.12 12 | 0.71 7 | 1.44 6 | 2.45 10 | 1.94 8 |
| $H_x$ (SPT) | 5.88 13 | 2.27 11 | 2.92 12 | 3.92 13 | 4.29 13 |
| (WSPT) | 0.36 6 | 0.31 5 | 1.48 7 | 1.02 6 | 2.21 10 |
| (LPT) | 0.98 9 | 1.22 8 | 4.33 13 | 2.64 11 | 3.14 12 |
| (WLPT) | 0.27 5 | 0.32 6 | 0.57 5 | 0.88 5 | 1.85 7 |
| ($W$) | 1.22 10 | 2.74 12 | 1.96 10 | 2.79 12 | 1.68 6 |

The numbers shown for each combination are (1) the difference between $\bar{F}_w$ and the lower bound and (2) the rank of the $\bar{F}_w$ value among the rules tested.

that no single heuristic will always be associated with the best schedule and that no single priority rule will always be associated with the best schedule.

Secondly, the rankings indicated quite clearly that the weighted priorities are more effective than their unweighted counterparts. Only under $H_x$ did a weighted priority lead to a rank below sixth.

Thirdly, $H_x$ did not produce a schedule ranked better than fifth and was clearly worse than the other heuristics. Presumably, $H_x$ suffers from the fact that it does not necessarily distribute the jobs in equal numbers among machines.

The heuristic $H_m$, by contrast, is restricted to distributing the jobs in equal numbers among machines. While this characteristic is sometimes favorable, it is distinctly unfavorable in the case of the unweighted priorities, which ranked 14 and 15 in all six problems.

Finally, there appears to be no overall clear choice between the priorities WSPT and WLPT. For $H_m$, WSPT seems to be uniformly more effective, but for $H_1$ no similar conclusion can be drawn. In some respects, this may be the most surprising property illustrated by these results, for although WLPT maximizes $\bar{F}_w$ for $m=1$, it can be incorporated into the parallel processor case in a desirable way.

The important results in the six data sets are summarized in Table 8 by the use of rankings where the specific rules which produced the three best schedules are shown in ranked order for all 30 problems. Of the 30 outcomes, the distribution of best schedules was as follows:

| | *Best* | *Second* | *Third* | *Total* |
|---|---|---|---|---|
| $H_1$ (WSPT) | 21 | 9 | 0 | 30 |
| $H_m$ (WSPT | 6 | 5 | 17 | 28 |
| $H_1$ (WLPT) | 3 | 16 | 8 | 27 |

TABLE 8. *Comparison of Rules*

| $m=$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| *DS*1 | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) |
| *DS*2 | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_m$ (WSPT)<br>$H_1$ (WLPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_m$ (WSPT)<br>$H_1$ (WLPT) |
| *DS*3 | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_m$ (WSPT)<br>$H_1$ (WLPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_1$ (WLPT)<br>$H_1$ (WSPT)<br>$H_m$ (WLPT) |
| *DS*4 | $H_m$ (WSPT)<br>$H_1$ (WSPT)<br>$H_m$ (WLPT) | $H_1$ (WLPT)<br>$H_1$ (WSPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_m$ (WSPT)<br>$H_1$ (WSPT)<br>$H_m$ (WLPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) |
| *DS*5 | $H_m$ (WSPT)<br>$H_1$ (WSPT)<br>$H_m$ (WLPT) | $H_1$ (WLPT)<br>$H_1$ (WSPT)<br>$H_m$ (WSPT) | $H_1$ (WSPT)<br>$H_m$ (WSPT)<br>$H_1$ (WLPT) | $H_1$ (WSPT)<br>$H_m$ (WSPT)<br>$H_1$ (WLPT) | $H_1$ (WSPT)<br>$H_m$ (WSPT)<br>$H_m$ (WLPT) |
| *DS*6 | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_m$ (WSPT)<br>$H_1$ (WSPT)<br>$H_1$ (WLPT) | $H_1$ (WSPT)<br>$H_1$ (WLPT)<br>$H_m$ (WSPT) | $H_m$ (WSPT)<br>$H_1$ (WSPT)<br>$H_1$ (WLPT) | $H_m$ (WSPT)<br>$H_1$ (WSPT)<br>$H_1$ (WLPT) |

$H_1$ (WSPT) most frequently produced the best schedule and always produced one of the two best schedules. $H_m$ (WSPT) and $H_1$ (WLPT) were less likely to produce the best schedule, but nearly as likely to produce one of the three best. While the size of the problem precludes a comparison of the best heuristic solution with the true optimum, we observed that $H_1$ (WSPT) was within 1 percent of the lower bound 95 percent of the time. Therefore, it appears that only very slight improvements could possibly be made over the solution obtained with this heuristic procedure.

## 5. CONCLUSIONS

If an optimum rule for this problem exists (that is, a scheduling mechanism more efficient than enumeration) it is likely to be quite complicated. Furthermore, extensions to flowtime problems with multiple resource types or with nonstatic job arrivals would also appear to be complex.

The primary element in attempting to minimize $\bar{F}_w$ with parallel processors is the use of the condition that WSPT should prevail for each processor. This condition is so important to near-optimal scheduling that only marginal improvements can be expected from sophisticated assignments of jobs to machines. Moreover, this investigation suggests that the relative behavior of heuristic procedures for this assignment process may be extremely difficult to characterize in general. The prospect is that special problem attributes (distribution of processing times, number of machines, etc.) will affect the performance of different procedures and perhaps render the concept of an "optimum rule" meaningless.

The results for the large job sets indicate that $H_x$ is the least effective of the three heuristics tested and that neither $H_1$ nor $H_m$ is consistently best. Indeed, for a given set of jobs, it is possible that the relative performance of $H_1$ and $H_m$ is reversed as $m$ is varied. Nevertheless, $H_1$ did appear to be perceptibly more likely than $H_m$ to produce the best schedule. The outcome is a pleasant surprise in that $H_1$ is the simplest of the three heuristics to implement and $H_x$ is the most difficult.

In much the same way, no priority ordering was consistently best, although it was clear that weighted priorites were more reliable than unweighted priorities. The effectiveness of WLPT might be attributable to the fact that longest-first sequencing tends to distribute processing fairly equally among machines, as discussed by Kedia [13] and illustrated in Figure 1. Nevertheless, WSPT appeared to be the best priority ordering.

The fact that both WSPT and WLPT were effective might suggest that weighting factors are more important job traits than processing times. Yet the largest-weight priority was unable to produce one of the three best schedules in any of the 30 problems.

With regard to the effect of processing time distributions and weighting factor distributions, the results are inconclusive. If anything, $H_1$ (WSPT) was most effective when the weights were uniformly distributed and was least effective for data sets 4 and 6, but these represented the job sets with the least and most variability. More testing would be required to determine whether there is a significant distribution effect. From the limited scope of these results, however, one might infer that the conclusions hold for a wide variety of distributions.

Granting the lack of consistency which is inherent in the problem, the data in Table 8 certainly recommend $H_1$ (WSPT) as the most effective scheduling procedure. In addition to the high frequency with which it produced good schedules, $H_1$ (WSPT) has other advantages. First, it is a logical rule to use, since it is a generalization of the optimum rule for the multiprocessor problem with equal weights. Secondly, it is a one-pass procedure, and does not require a reordering of the jobs once they have been assigned to machines. It is slightly simpler than $H_m$ (WSPT), which includes an additional assignment mechanism at each stage, and is probably the simplest procedure of those studied. Finally, $H_1$ (WSPT)

structurally is a dispatching procedure: the final job assignments are made at chronologically ordered points in time (i.e., in the order they would be implemented.) This type of structure is likely to be more adaptable as part of a larger, more complex problem than two-pass procedures or iterative schemes. In particular, problems with multiple resource types or with dynamic job arrivals are important extensions of the problem considered here, and they can accomodate the $H_1$ (WSPT) heuristic without major obstacles.

## BIBLIOGRAPHY

[1] Abramowitz, M. and I. A. Stegun, *Handbook of Mathematical Functions*, National Bureau of Standards, Applied Mathematics Series, (1964), pp. 824–825.

[2] Baker, N. R., "Optimal User Search Sequences and Implications for Information System Operation," American Documentation, Vol. 20, No. 3. (July 1969), pp. 203–212.

[3] Bowdon, E. K., "Priority Assignment in a Network of Computer," IEEE Transactions on Computers, C–18, 11 (Nov. 1969), pp. 1021–26.

[4] Coffman, E. G. and R. R. Muntz, "Models of Pure Time-Sharing Disciplines for Resource Allocations," Proc. 24th ACM National Conference (1969), pp. 217–228.

[5] Conway, R. W., W. L. Maxwell and L. W. Miller, *Theory of Scheduling* (Addison-Wesley, Reading, Mass., 1970).

[6] Denby, D. C., "Minimum Downtime as a Function of Reliability and Priority Assignments in Component Repair," Journal of Industrial Engineering Vol. 17, No. 7 (1967), pp. 436–439.

[7] Eastman, W. L., S. Even and I. M. Isaacs, "Bounds for the Optimal Scheduling of $n$ Jobs on $m$ Processors," Management Science Vol. 11, No. 2 (Nov. 1964), pp. 268–279.

[8] Feller, W. "An Introduction to Probability Theory and Its Applications" (John Wiley & Sons, New York, 1957), Vol. 1.

[9] Gapp, W., P. S. Mankekar and L. G. Mitten, "Sequencing Operations to Minimize In-process Inventory Costs," Management Science Vol. 11, No. 3 (Jan. 1965), 476–484.

[10] Grieshop, D. S. "An Analysis of the System Effectiveness of a Sequential Manpower Training Model," M.S. Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology (Jan. 1972).

[11] Held, M. and R. M. Karp, "A Dynamic Programmic Approach to Sequencing Problems," Journal of SIAM Vol. 10, No. 1 (Mar. 1962), pp. 196–210.

[12] Hu, T. C. "Parallel Sequencing and Assembly Line Problems," Operations Research Vol. 9, No. 6 (Nov. 1961), pp. 841–848.

[13] Kedia, S. K., "A Job Shop Scheduling Problem with Parallel Machines," Technical Report 71–6, Department of Industrial Engineering, University of Michigan (1971).

[14] Knuth, D. E., *The Art of Computer Programming: Fundamental Algorithms* (Addison-Wesley, Reading, Massachusetts, 1968) pp. 51–73.

[15] McNaughton, R., "Scheduling with Deadlines and Loss Functions," Management Science, Vol. 6, No. 1 (Oct. 1959), pp. 1–12.

[16] Maxwell, W. L., "On the Generality of the Equation $L=\lambda W$," Operations Research Vol. 18, No. 1 (Jan.–Feb. 1970), pp. 172–174.

[17] Merten, A. G., "Some Quantitative Techniques for File Organizations," Technical Report No. 15, University of Wisconsin Computing Center (June 1970).

[18] Mitten, L. G., "An Analytic Solution to the Least-Cost Testing Sequence Problem," Journal of Industrial Engineering, Vol. 11, No. 1 (Jan.–Feb. 1960), p. 17.

[19] Muntz, R. and E. Coffman, "Optimal Pre-emptive Scheduling on Two-Processor Systems," IEEE Transactions on Computers, Vol. C–18, No. 11 (Nov. 1969), pp. 1017–1020.

[20] Price, H. W., "Least-Cost Testing Sequence," Journal of Industrial Engineering, Vol. 10, No. 4, (July–Aug. 1959), pp. 278–279.

[21] Rau, J. G., "Minimizing a Function of Permutations of $n$ Integers," Operations Research, Vol. 19, No. 1, (Jan.–Feb. 1971), pp. 237–240.

[22] Riesel, H. "In which Order are Different Conditions to be Examined?," *BIT*, Vol. 3 (1963), pp. 255–256.

[23] Rothkopf, M., "Scheduling Independent Tasks on Parallel Processors," Management Science, Vol. 12, No. 5 (Jan. 1966), pp. 437–447.

[24] Smith, W. E., "Various Optimizers for Single-Stage Production," Nav. Res. Log. Quart. *3*, 59–66 (1956).