

Solving Machine Shops Scheduling Problems using Priority Sequencing Rules Techniques

Engr. Dr. A.C. Uzorh^{1*} and Nnanna Innocent^{2*}

^{*1}Department of Mechanical Engineering, Federal University of Technology Owerri, Imo State

^{*2}Department of Mechanical Engineering Akanu Ibiam Federal Polytechnic Unwana, Ebonyi State Nigeria

ABSTRACT

This paper considers a scheduling problem with component availability constraints in a machine shop of only one engine expert. The instruments used were personal interview and observations along the production line. Two priority sequencing rules were compared in an extensive numerical study. Results show that the Earliest Due Date (EDD) rule is better than the Shortest Processing Time (SPT) rule with respect to average past due (keeping promises to customers), but worse with respect to average flow time for the set of jobs in this study. Also EDD schedule gave better customer service, as measured by the average hours past due, and a lower maximum hours past due (22 versus 24). However, the SPT schedule provided a lower average flow time. In general, the SPT priority rule will push most jobs through the system to completion more quickly than will the other rules. Speed can be an advantage—but only if jobs can be delivered sooner than promised and revenue collected earlier. If they cannot, the completed job must stay in finished inventory. Consequently, the priority rule chosen can help or hinder the firm in meeting its competitive priorities, therefore, management's choice depends on which performance measure it values the most. More experimentation should be conducted before a final choice is made.

KEY WORDS: Job scheduling, Machine Shop, processing time, Optimization scheduling

Date of Submission: 14 May 2014



Date of Publication: 10 June 2014

I. INTRODUCTION

Scheduling is the allocation of shared resources over time to competing activities. It can also be defined as prescribing of when and where each operation necessary to manufacture a product is to be performed. The principle aim of scheduling is to plan the sequence of work so that production can be systematically arranged towards the end of completion of all products by due date (Suresh and Chaudhuri, 1993). Emphasis has been on investigating *machine scheduling problems* where *jobs* represent activities and *machines* represent resources; each machine can process at most one job at a time. This paper was motivated by the current scheduling problem of Engr. Oformba Machine Shop located at No23 Anukwuru street owerri, Imo state. The machine shop presently rebore engine blocks. However, five engine blocks are waiting for processing. At any time, the company has only one engine expert on duty who can do this type of work. The engine problems have been diagnosed, and the processing times for the jobs have been estimated. Expected completion times have been agreed upon with the shop's customers. The workshop open's from 8:00 A.M. until 5:00 P.M. each weekday, plus weekend hours as needed, the customer pickup times are measured in business hours from the current time. The management is faced with this new challenge because of the number of the engine expert available. Considering this situation this paper developed two scheduling decision rules namely: (a) the EDD rule and (b) the SPT rule. The essence is to compare the EDD and SPT rules and finally arrived at acceptable decision point.

1.1 The process of scheduling

Finite-capacity scheduling systems are available for virtually all types of production environments—job-shop, repetitive, repetitive-batch, continuous-process, process-batch, and mixed mode. Obviously, it is important to select a system specifically suited to the manufacturing mission of the plant to be scheduled. Two major categories of systems are available: single plant and multiple-plant scheduling (Suresh and Chaudhuri, 1993). Multi-plant systems are considerably more complex and are best considered as extended planning systems since they involve high-level (MPS) allocation of orders to multiple facilities. Plant scheduling systems usually use one of four basic methods of processing orders through the plant. Each method involves modeling the plant, each schedules to finite capacity, and each uses one or more prioritization rule(s).

For practical purposes, each method may be thought of as a simulation of how orders would be processed through the production resources of the plant given a certain value judgment regarding the most important objective to be achieved (Kiran and Thomas, 1992).

Job scheduling has as its primary objective maximization of opportunity for the most important orders to be completed on time. Jobs are scheduled through all their operations in priority sequence, in effect anticipating when capacity will be needed for high-priority orders and showing what effect the arrival of those orders will have on future queuing at work centers (Pinedo and Chao, 1998).

Job scheduling is comparatively easy to implement, easy to understand, and fast for computer processing. The theoretical concern that gaps in the schedule can cause long cycle times for some jobs seldom materializes when the system is properly used.

Resource scheduling is based on the theory of constraints, which mandates that bottleneck resources must be completely utilized. Predetermined bottlenecks are scheduled first with all operations requiring them. Then, remaining operations of each order are scheduled both backward and forward from the bottleneck. The first pass at scheduling a designated bottleneck may create overloads at noncritical work centers, requiring iterations of the backward/forward process, which consumes computer time. Consequently, resource scheduling works best in environments having few bottlenecks that do not shift between work centers.

Event scheduling uses clock-based simulation to schedule each queue at each work center on an individual basis. Time resolution is usually very fine, with the clock advancing until completion of an activity permits another activity to commence. Event scheduling usually produces good schedules, but may require long computer processing time in complex environments.

Optimization scheduling seeks to optimize a user-perceived value. Such systems have been described as “optimal seeking, but do not guarantee an optimal solution.” Such systems are seductively dangerous, inasmuch as they appear to give the user what is asked for, no matter how wrongheaded or shortsighted the objective. For example, a schedule that optimizes short-range profit margins may have a very adverse effect on the business as a whole (Vollmann, et al, 2005). Optimization scheduling requires long computer processing times under real-world conditions, so they are not always well suited for the dynamics of routine production control. Many systems purport to be finite-capacity scheduling when in reality they are little more than computer-based manual scheduling boards (LaForge, 2000). Users with serious intentions of reliable and efficient scheduling should ascertain before selection that the software under consideration employs one of the four methods described here. Within this group of four, different approaches are used by different designers, with varying degrees of elegance of simulation. The most elegant are not always the most useful, so it is important to select a system practical for operating people to use to support decisions and actions in the fast-paced, dynamic, daily routine of production control (Hartvigsen, 2004).

1.2 Forward Scheduling

This is commonly used in job shops where customers place their orders on needed as soon as possible basis. It determines start and finish times of next priority job by assigning it the earliest available time slot and from that time, determines when job will be finished in that work centre. Since the job and its components start as early as possible, they will typically be completed before they are due at the subsequent work centres in the routing.

1.3 Backward Scheduling

This is often used in assembly type industries and commit in advance to specific delivery dates. Backward scheduling determines the start and finish times for waiting jobs by assigning them to the latest available time slot that will enable each job to be completed just when it is due, but not before. Backward scheduling minimizes inventories, by assigning jobs as late as possible.

It is important to note that forward scheduling gets jobs done in shorter lead times, compared to backward scheduling. Figure 1.0 shows forward and backward scheduling.

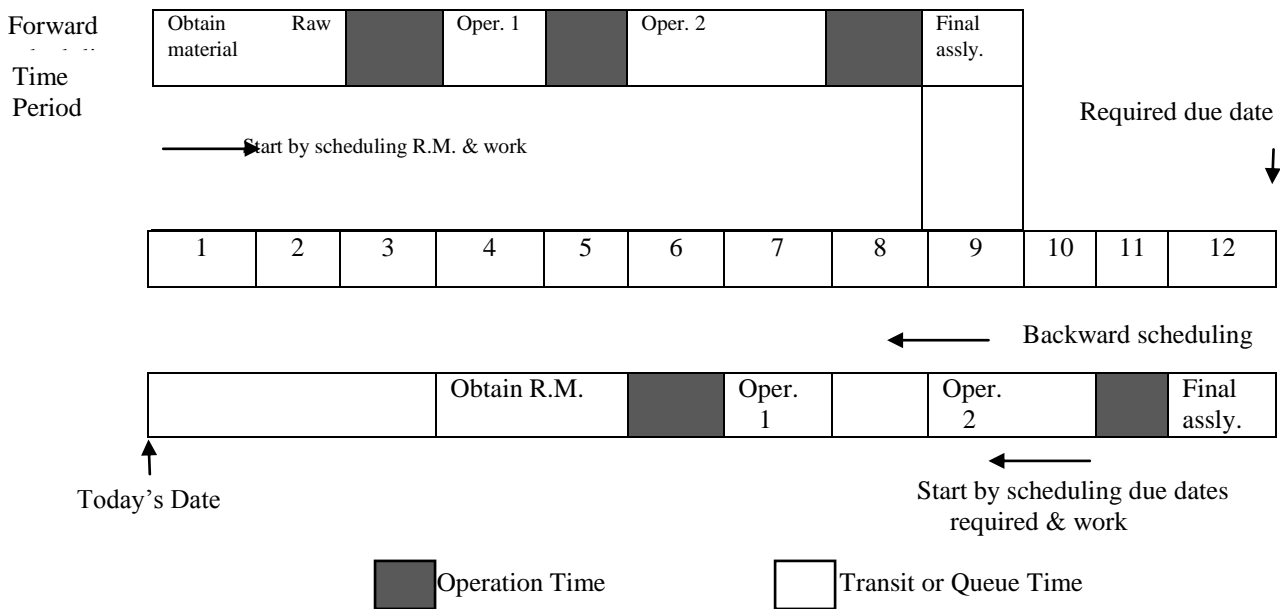


Figure 1.0: Forward and backward scheduling (Metters and Vincente, 1999)

II. JOB SCHEDULING PROBLEM

A work location in which a number of general purpose work stations exist and are used to perform a variety of jobs. Example: Car repair – each operator (mechanic) evaluates plus schedules, gets material, etc. – Traditional machine shop, with similar machine type located together, batch or individual production. The short-term schedules show an optimal order (sequence) and time in which jobs are processed. They also show timetables for jobs, equipment, people, materials, facilities and all other resources that are needed to support the production plan. The schedules should use the resources efficiently to give low costs and high utilizations. Other purpose of scheduling are, minimizing customers wait time for a product, meeting promised delivery dates, keeping stock levels low, giving preferred working patten, minimizing waiting time of patients in a hospital for different types of tests, and so on (Baker, 2002). The sequencing problem is the problem of finding an optimal sequence of completing certain number of jobs so as to minimize the total elapsed time between completion of first and last job. The general scheduling or sequencing problem may be described as: Let there be a jobs to be performed, one at a time, on each or at machines. The sequence (order) of the machines in which each job should be performed is given. The actual or expected time required by the jobs on each of the machine is also given. The general sequencing problem, therefore, is to find the sequence out of $(n!)^m$ possible sequences, which minimize the total elapsed time between the start of the job in the first machine and the completion of the last job on the last machine.

In particular if, $n = 3$ and $m = 3$, then the total number of possible sequences will be $(3!)^3 = 216$. Theoretically, it may be possible to find the, optimum sequence but this would require a lot of computational time. Thus, one should adopt the sequencing technique.

To find the optimum sequence, we first need to calculate the total elapsed time for each of the possible sequences. As stated earlier, even if the values of m and n are very small, it is difficult to get the desired sequence with the total minimum elapsed time. However, due to certain rules designed by Johnson, the task of determining an optimum sequence has become quite easy (Johnson, 1954).

2.1 Notations

Notations

t_{ij} = Processing time (time required) for job i on machine j .

T = Total elapsed time for processing all the jobs. This includes idle time, if any.

i_{ij} = Idle time on machine j from the end of job $(i-1)$ to the start of job i .

2.2 Scheduling procedures

• First, break the problem into the following categories:

1. N jobs, 1 machine
2. N jobs, 2 machines (flow shop)
3. N jobs, 2 machines (any order)
4. N jobs, 3 machines (flow shop)
5. N jobs, M machines

Scenario 1 – n jobs, 1 machine (I)

• Let P_1, P_2, \dots, P_n be processing time for each job – (including setup)

- The schedule possibilities are the permutations of n, which is equal to “n!”
- Since the total processing time, or make-span is independent of sequence, this is not a criterion for choice – Consider using minimum mean flow time

Scenario 1 – n jobs, 1 machine (II)

Flow time for job in kth position is:

$$\bar{F}_{[k]} = \frac{\sum_{i=1}^k P_{[i]}}{n}$$

Mean flow time for n jobs:

$$\bar{F} = \frac{\sum_{k=1}^n \bar{F}_{[k]}}{n} = \frac{\sum_{k=1}^n \sum_{i=1}^k P_i}{n}$$

$$\bar{F} = \frac{\sum_{i=1}^n (n-i+1)P_i}{n}$$

Scenario 1 – n jobs, 1 machine (III)

It can be proven that \bar{F} is minimized by taking jobs in order of shortest processing time [SPT]

That is order by increasing P, so that

$$P_{[1]} \leq P_{[2]} \leq P_{[3]} \dots \leq P_{[n]}$$

Scenario 1 – n jobs, 1 machine (IV)

Provide numerical weighting to jobs by priority

(w) – higher w, more important then

$$\bar{F}_w = \frac{\sum_{i=1}^n W_i P_{[i]}}{n}$$

and jobs are sequenced by:

$$\frac{P_{[1]}}{W_{[1]}} \leq \frac{P_{[2]}}{W_{[2]}} \leq \frac{P_{[3]}}{W_{[3]}} \leq \dots \leq \frac{P_{[n]}}{W_{[n]}}$$

N Jobs, M Machines

Number of possible schedules is extremely large, (n!)^m

Almost all solved by heuristics which are based on sequencing or dispatching rules.

N Jobs, M Machines

List of Heuristics are as follows:

1. R (Random) – Pick any Job in Queue with equal probability. This rule is often used as benchmark for other rules
 2. FCFS (First Come First Serve) – Jobs are processed in the order in which they arrived at the work center (also called earliest release date)
 3. SPT (Shortest Processing Time) – This rule tends to reduce both work-in-process inventory, the average job completion (flow) time, and average job lateness.
 4. EDD (Earliest Due Date) – Choose Job that has earliest due date
 5. CR (Critical Ratio) = Processing Time / Time until due (Due Date – Current Time).
Take the highest value.
 6. LWR (Least Work Remaining) – This rule is an extension of SPT variant that considers the number of successive operations
 7. ST (Slack Time) = Time until job is due - (Sum of processing time remaining). Take the job with the smallest amount of slack time.
 8. ST/O (Slack Time per Remaining Operation) = slack time divided by number of operations remaining. Take the job with the smallest amount of slack time per remaining operation
- When in Doubt, use SPT. Also, use SPT to break ties.

2.3 Processing n jobs through two machines

Let there be n jobs, each of which are to be processed through two machines, M₁ and M₂ in the order M₁M₂, i.e. each job has to pass through the same sequence of operations. In other words, a job is assigned on machine M₁, first and after it has been completely processed on machine M₁, it is assigned on machine M₂. If the machine M₂, is not free at the moment for processing the same job, then the job has to wait in a waiting line for its turn on machine M₂, i.e. passing is not allowed.

Since passing is not allowed, therefore, machine M₁ will remain busy in processing all the n jobs one- by-one, while machine M₂, may remain idle time of the second machine. This can be achieved only by determining

sequence of n jobs, which are to be processed on two machines M_1 and M_2 . The procedure suggested by Johnson for determining the optimal sequence can be summarized as follows (Pinedo, 2002):

The Algorithm

Step 1: List the jobs along with their processing times on each machine in a table, as shown below:

Processing time on machine	Job number					
	1	2	3	4	...	n
M_1	t_{11}	t_{12}	t_{13}	t_{14}	...	t_{1n}
M_2	t_{21}	t_{22}	t_{23}	t_{24}	...	t_{2n}

Step 2: Examine the columns for processing times on machines M_1 and M_2 and find the shortest processing time in each column, i.e. find out, $\min(t_{1j}, t_{2j})$ for all j.

Step 3 (a): If the shortest processing time is on machine M_1 , then schedule the job as early as possible, without moving jobs already scheduled, i.e. place the job in the first available position in the sequence. If the processing time is on machine M_2 , then schedule the job as late as possible, without moving any jobs already scheduled, i.e. place the job in the last available position in the sequence.

b. If there is a tie in selecting the minimum of all the processing times, then the following three situations may arise:

(i) Minimum among all processing times is same for the machine, i.e. $\min(t_{1j}, t_{2j}) = t_{1k} = t_{2r}$, then process the kth job first and the rth job last.

(ii) If the tie for the minimum occurs among processing times t_{1j} on machine M_1 only, then select the job corresponding to the smallest job subscript first.

(iii) If the tie for the minimum occurs among processing times t_{2j} on machine M_2 , then select the job corresponding to the largest job corresponding to the largest job subscript last.

Step 4: Remove the assigned jobs from the table. If the table is empty, stop and go to Step 5. Otherwise, go to Step 2.

Step 5: Calculate the idle time for machines M_1 and M_2 :

a. Idle time for machine $M_1 =$ (Total elapsed time) — (Time when the last job in a sequence finishes on machine M_1)

b. Idle time for machine $M_2 =$ Time at which the first job in a sequence finishes on machine $M_1 + \sum_{j=2}^n \{ \text{Time when the } j\text{th job in a sequence starts on machine } M_2 \} - \{ \text{Time when the } (j-1)\text{th job in a sequence finishes on machine } M_2 \}$

Step 6: The total elapsed time to process all jobs through two machines is given by.

$$\begin{aligned} \text{Total elapsed time} &= \text{Time when the } n\text{th job in a sequence finishes on machine } M_2 \\ &= \sum_{j=1}^n M_{2j} = \sum_{j=1}^n I_{2j} \end{aligned}$$

Where $M_{2j} =$ Time required for processing jth job on machine M_2 .

$t_{2j} =$ time for which machine M_2 remains idle after processing (j-1)th job and before starting work in jth job.

2.4 Processing n jobs through three machines

Johnson provides an extension of his procedure to the case in which there are three instead of two machines. Each job is to be processed through three machines $M_1, M_2,$ and M_3 in the order M_1, M_2, M_3 . The list of jobs with their processing times is given below. An optimal solution to this problem can be obtained if either or both of the following conditions hold good (Johnson, 1954):

Processing time on machine	Job number					
	1	2	3	4	...	n
M_1	t_{11}	t_{12}	t_{13}	t_{14}	...	t_{1n}
M_2	t_{21}	t_{22}	t_{23}	t_{24}	...	t_{2n}
M_3	t_{31}	t_{32}	t_{33}	t_{34}	...	t_{3n}

The minimum processing time on machine M_1 is at least as great as the maximum processing time on machine M_2 . that is, $\min t_{1j} \geq \max t_{2j}$ for $j = 1, 2, \dots, n$

The minimum processing time on machine M_3 is at least as great as the maximum processing time on machine M_2 that is $\min t_2 \geq \max t_{2j}$ for $j = 1, 2, \dots, n$

If either or both the above conditions hold good, then the steps of the algorithm can be summarized in the following steps:

The Algorithm

Step 1: Examine the processing times of the given jobs on all three machines and if either one or both the above conditions hold, then go to Step 2, otherwise the algorithm fails.

Step 2: Introduce two fictitious machines, say G and H with corresponding processing times given by:

$t_{Gj} = t_{1j} + t_{2j}$ $j = 1, 2, \dots, n$ that is, the processing time on machine G is the sum of the processing times on machines M_1 and M_2 and

$t_{Hj} = t_{2j} + t_{3j}$ $j = 1, 2, \dots, n$ that is, processing time on machine H is the sum of the processing times on machines M_2 and M_3

Step 3: Determine the optimal sequence of jobs for this n-job, two machine equivalent sequencing problem with the prescribed ordering GH in the same way as discussed earlier.

2.5 Processing n jobs through M machines

Let there be n jobs, each of which is to be processed through m machines, say M_1, M_2, \dots, M_m in the order $M_1M_2 \dots M_m$. The optimal solution to this problem can be obtained if either or both of the following conditions hold good.

- (a) $\text{Min} \{t_{1j}\} \geq \text{Max} \{t_{ij}\}; j = 2, 3, \dots, m - 1$ and/or
- (b) $\text{Min} \{t_{mj}\} \geq \text{Max} \{t_{ij}\}; j = 2, 3, \dots, m - 1$

That is, the minimum processing time on machines M_1 and M_m is as great as the maximum processing time on any of the remaining $(m - 1)$ machines.

If either or both these conditions hold well, then the steps of the algorithm can be summarized in the following steps:

Step 1: Find, $\text{Min} \{t_{1j}\}$, $\text{Min} \{t_{mj}\}$ and $\text{Max} \{t_{ij}\}$ and verify the above conditions. If either or both the conditions mentioned above hold, then go to step 2. Otherwise the algorithm fails.

Step 2: Convert m-machine problem into 2-machine problem by introducing two fictitious machines, say G and H with corresponding processing time given by:

$$(i) \quad t_{Gj} = t_{1j} + t_{2j} + \dots + t_{m-1j}; \quad j = 1, 2, \dots, n$$

i.e. processing time of n-jobs on machine G is the sum of the processing times on machines M_1, M_2, \dots, M_{m-1}

$$(ii) \quad t_{Hj} = t_{2j} + t_{3j} \dots + t_{mj}; \quad j = 1, 2, \dots, n$$

i.e. processing time of n-jobs on machines M_2, M_3, \dots, M_m

Step 3: the new processing times obtained, can now be used for solving n-job, two-machine equivalent sequencing problem with the prescribed ordering HG in the same way as discussed earlier.

Remarks

1. In addition to the condition given in Step 2 if:

$$t_{2j} + t_{3j} \dots + t_{m-1j} = k \text{ (constant)}$$

for all $j = 1, 2, \dots, m - 1$, then the optimal sequence can be obtained for n-jobs and two machines M_1 and M_m in the order M_1M_m as usual.

2. If $t_{1j} = t_{mj}$ and $t_{Gj} = t_{Hj}$, for all $j = 1, 2, \dots, n$, then the total number of optimal sequences will be n and total minimum elapsed time in these cases would also be the same.

3. The method described above for solving 'i-jobs and in-machines sequencing problem is not a general method. It is applicable only to certain problems where the minimum cost (or time) of processing the jobs through first and/or last machine is more than or equal to the cost (or time) of processing the jobs through the remaining machines.

III. METHODOLOGY

A few visits were organized to the firm to study their production line layout (workstation) and process flows. The following conditions as stated in table 4.1 were obtained.

Table 4.1: Product processing the current situation of the firm

Engine Block	Business Hours Since Order Arrived	Processing Time, Including Setup (hours)	Business Hours Until Due Date (Customer pickup time)
Ranger	12	8	10
Explorer	10	6	12
Bronco	1	15	20
Econoline 150	3	3	18
Thunderbird	0	12	22

IV. DATA ANALYSIS

Two priority sequencing rules approaches namely: (1) earliest due date (EDD) and shortest processing time (SPT) were adopted to analysis and compare the results.

Earliest due date (EDD) Approach

The earliest due date rule states that the first engine block in the sequence is the one with the closest due date. From table 4.1, the Ranger engine block is processed first. The Thunderbird engine block, with its due date furthest in the future, is processed last. The sequence is shown in the following table 5.1, along with the flow times, the hours early, and the hours past due.

Table 5.1: Sequence of operation using approach 1

Engine Block Sequence	Hours Since Order Arrived	Begin Work		Processing Time (hr)		Finish Time (hr)	Flow Time (hr)	Scheduled Customer Pickup Time	Actual Customer Pickup Time	Hours Early	Hours Past Due
Ranger	12	0	+	8	=	8	20	10	10	2	-
Explorer	10	8	+	6	=	14	24	12	14	-	2
Econoline 150	3	14	+	3	=	17	20	18	18	1	-
Bronco	1	17	+	15	=	32	33	20	32	-	12
Thunderbird	0	32	+	12	=	44	44	22	44	-	22

From table 5.1, the flow time for each job is its finish time, plus the time since the job arrived. For example, the Explorer engine block's finish time will be 14 hours from now (8 hours waiting time before the engine expert started to work on it plus 6 hours processing). Adding the 10 hours since the order arrived at this workstation (before the processing of this group of orders began) results in a flow time of 24 hours. You might think of the sum of flow times as the total job hours spent by the engine blocks since their orders arrived at the workstation until they were processed.

Thus, the performance measures for the EDD schedule for the five engine blocks are

$$\text{Average flow time} = \frac{20+24+20+33+44}{5} = 28.2 \text{ hrs}$$

$$\text{Average hours early} = \frac{2+0+1+0+0}{5} = 0.6 \text{ hrs}$$

$$\text{Average hours past due} = \frac{0+2+0+12+22}{5} = 7.2 \text{ hrs}$$

Shortest processing time (SPT)

From table 4.1, under the SPT rule, the sequence starts with the engine block that has the shortest processing time, the Econoline 150, and it ends with the engine block that has the longest processing time, the Bronco.

The sequence, along with the flow times, early hours, and past due hours, is contained in table 5.2.

Table 5.2: sequence operation under Scenario 2

Engine Block Sequence	Hours Since Order Arrived	Begin Work		Processing Time (hr)		Finish Time (hr)	Flow Time (hr)	Scheduled Customer Pickup Time	Actual Customer Pickup Time	Hours Early	Hours Past Due
Econoline 150	3	0	+	3	=	3	6	18	18	15	-
Explorer	10	3	+	6	=	9	19	12	12	3	-
Ranger	12	9	+	8	=	17	29	10	17	-	7
Thunderbird	0	17	+	12	=	29	29	22	29	-	7
Bronco	1	29	+	15	=	44	45	20	44	-	24

The performance measures are

$$\text{Average flow time} = \frac{6+19+29+29+45}{5} = 25.6 \text{ hrs}$$

$$\text{Average hours early} = \frac{15+3+0+0+0}{5} = 3.6 \text{ hrs}$$

$$\text{Average hours past due} = \frac{0+0+7+7+24}{5} = 7.6 \text{ hrs}$$

V. RESULTS AND DISCUSSIONS

The EDD rule is better than the SPT rule with respect to average past due (keeping promises to customers), but worse with respect to average flow time for the set of jobs in this study. Management's choice depends on which performance measure it values the most. More experimentation should be conducted before a final choice is made. As the analysis of study shows, the EDD schedule gave better customer service, as measured by the average hours past due, and a lower maximum hours past due (22 versus 24). However, the SPT schedule provided a lower average flow time. In general, the SPT priority rule will push most jobs through the system to completion more quickly than will the other rules. Speed can

be an advantage—but only if jobs can be delivered sooner than promised and revenue collected earlier. If they cannot, the completed job must stay in finished inventory. Consequently, the priority rule chosen can help or hinder the firm in meeting its competitive priorities. The EDD rule performs well with respect to the percentage of jobs past due and the variance of hours past due. For any set of jobs to be processed at a single workstation, it minimizes the maximum of the past due hours of any job in the set. The EDD rule is popular with firms that are sensitive to achieving due dates, which usually are the basis for setting priorities using MRP systems. The SPT rule tends to minimize the mean flow time (assuming time since arrival is 0 for all jobs) and the percentage of jobs past due. It also tends to maximize shop utilization. For the single-workstation case, the SPT rule always will provide the lowest mean finish time. However, it could increase total inventory because it tends to push all work to the finished state. In addition, it tends to produce a large variance in past due hours because the larger jobs might have to wait a long time for processing. Also, it provides no opportunity to adjust schedules when due dates change. The advantage of this rule over others diminishes as the load on the shop increases.

VI. CONCLUSION AND RECOMMENDATIONS

The challenges faced by management of Engr. Oformba machine shop has been examined in this paper with the view that, oneway to generate schedules in machine shops is by usingpriority sequencing rules,which allows the schedule for a workstation to evolve over a periodof time. The decision about which job to process next is made with simple priority ruleswhenever the workstation becomes available for further processing. One advantage of thismethod is that last-minute information on operating conditions can be incorporated intothe schedule as it evolves. Finally, we recommend that though the FCFS rule is considered fair to the jobs (or customers), it performs poorly with respect to all performance measures. This result is to be expected because FCFS does not acknowledge any job (or customer) characteristics. However, FCFS usually is the only acceptable choice for service processes where the customer is present and demand leveling options such as appointments or reservations are not used.

REFERENCE

- [1] Baker, K. R., (2002). *Elements of Sequencing and Scheduling*. Hanover, NH: Baker Press.
- [2] Hartvigsen, D., (2004). *Sim Quick: Process Simulation with Excel*, 2nd ed. Upper Saddle River, NJ: Prentice Hall.
- [3] Johnson, S. M., (1954). Optimal Two Stage and Three Stage Production Schedules with Setup Times Included. *Naval Logistics Quarterly*, vol. 1, no. 1 (1954), pp. 61–68.
- [4] Kiran, A. S., and Thomas H. W., (1992). Simulation: Help for Your Scheduling Problems. APICS— *The Performance Advantage*, pp. 26–28.
- [5] LaForge, R. L., and Christopher W. C., (2000). Computer-Based Scheduling in Manufacturing Firms: Some Indicators of Successful Practice. *Production and Inventory Management Journal*, pp. 29–34.
- [6] Metters, R. and Vincente, V., (1999). A Comparison of Production Scheduling Policies on Costs, Service Levels, and Schedule Changes. *Production and Operations Management*, vol. 17, no. 3, pp. 76–91.
- [7] Pinedo, M., (2002) *Scheduling: Theory, Algorithms, and Systems*, 2nd ed. Upper Saddle River, NJ: Prentice Hall.
- [8] Pinedo, M., and Chao, X., (1998). *Operations Scheduling with Applications in Manufacturing and Services*. Boston: McGraw-Hill/Irwin.
- [9] Suresh, V., and Chaudhuri, D., (1993). Dynamic Scheduling—A Survey of Research. *International Journal of Production Economics*, vol. 32, pp. 52–63.
- [10] Vollmann, T. E., William, B., Whybark, D. C. and Robert, J., (2005). *Manufacturing Planning and Control Systems for Supply Chain Management*, 5th ed. New York: McGraw-Hill/Irwin.