Scheduling of Re-Entrant Flow Shops

Stephen C. Graves
Harlan C. Meal
Daniel Stefek
Abdel Hamid Zeghmi

WP#1438-83 May 1983

SCHEDULING OF RE-ENTRANT FLOW SHOPS

Stephen C. Graves, Harlan C Meal, Daniel Stefek and Abdel Hamid Zeghmi

Sloan School of Management

Massachusetts Institute of Technology

Cambridge, MA 02139

October, 1982

revised May 1983

Executive Summary

We propose and develop a scheduling system for a very special type of flow shop. This flow shop processes a variety of jobs that are identical from a processing point of view. All jobs have the same routing over the facilities of the shop and require the same amount of processing time at each facility. Individual jobs, though, may differ since they may have different tasks performed upon them at a particular facility. Examples of such shops are flexible machining systems and integrated circuit fabrication processes. In a flexible machining system, all jobs may have the same routing over the facilities, but the actual tasks performed may differ; for instance, a drilling operation may vary in the placement or size of the holes. Similarly, for integrated circuit manufacturing, although all jobs may follow the same routing, the jobs will be differentiated at the photolithographic operations. The photolithographic process establishes

patterns upon the silicon wafers where the patterns differ according to the mask that is used.

The flow shop that we consider has another important feature, namely the job routing is such that a job may return one or more times to any facility. We say that when a job returns to a facility it reenters the flow at that facility, and consequently we call the shop a re-entrant flow shop. In integrated circuit manufacturing, a particular integrated circuit will return several times to the photolithographic process in order to place several layers of patterns on the wafer. Similarly, in a flexible machining system, a job may have to return to a particular station several times for additional metal-cutting operations.

These re-entrant flow shops are usually operated and scheduled as general job shops, ignoring the inherent structure of the shop flow. Viewing such shops as job shops means using myopic scheduling rules to sequence jobs at each facility and usually requires large queues of work-in-process inventory in order to maintain high facility utilization, but at the expense of long throughput times.

In this paper we develop a cyclic scheduling method that takes advantage of the flow character of the process. The cycle period is the inverse of the desired production rate (jobs per day). The cyclic schedule is predicated upon the requirement that during each cycle the shop should perform all of the tasks required to complete a job, although possibly on different jobs. In other words, during a cycle period we require each facility to do each task assigned to it exactly once. With this requirement, a cyclic schedule is just the sequencing and timing on each facility of all of the tasks that that facility must perform during each cycle period. This cyclic schedule is to be repeated by each facility each cycle period. The determination of the best cyclic schedule is a very difficult combinatorial optimization problem that we cannot solve optimally for actual operations. Rather, we present a computerized heuristic procedure that seems very effective at producing good schedules. We have found that the throughput time of these schedules is much less than that achievable with myopic sequencing rules as used in a job shop. We are attempting to implement the scheduling system at an integrated circuit fabrication facility.

I. Introduction

This paper describes a new class of manufacturing operations, not described previously in the operations management literature. It also describes a simple and effective approach to the scheduling of these operations.

These manufacturing operations process jobs or lots, all of which are identical from a processing standpoint. Each job requires the same number of tasks (operations or processing steps) and corresponding tasks on succeeding jobs are performed on the same facility and require the same processing time. A flow shop such as an automobile assembly line is one example. The shop considered here has this characteristic and one further important feature.

We consider a shop which processes jobs that may return to the same facility or machine one or more times before completion. Flexible machining systems are often like this. In integrated circuit (IC) manufacturing the silicon wafers go through the same photolithographic steps several times as different levels in the integrated circuit chip are prepared.

These processes cannot be treated as simple flow shops. The repetitive use of the same facilities by the same job means that there may be conflicts among jobs, at some facilities, at different stages in the process. Later tasks to be done on a particular job by some facility may interfere with earlier tasks to be done at the same facility on a job which started later. This "re-entrant" or returning characteristic makes the process look more like a job shop on first examination. Jobs arrive at a facility from several different sources or predecessor facilities and may go to several successor facilities. However, there is much more structure than in the traditional job shop or flow shop, since all jobs are identical from a processing standpoint. In this sense the operation resembles an assembly line.

Such shops can be operated as though they were job shops, using myopic sequencing or local dispatch rules at each facility. This tends to lead to substantial job throughput times and accompanying inprocess inventory because arrivals at any facility are not scheduled but are allowed to be variable and unpredictable. Since the jobs to be processed are all alike or nearly so (successive lots of integrated circuits may differ only in the photographic patterns used, the processing steps are identical) it seems that a deterministic schedule could be found that makes good use of facility capacity without causing a large increase in throughput time.

The motivation for this work came from a plant manufacturing integrated circuits. In one shop which converts silicon wafers to chips ready for packaging there are 69 facilities which perform a total of 185 tasks in the course of completing one wafer lot. Note that this implies an average of more than two tasks per facility. Indeed, a lot returns to some facilities as many as eight times in this shop. While it is conceptually feasible to prepare manually a Gantt chart schedule for this process, it is not practical and we would not expect very good schedules to result.

Prior to the development of the scheduling method described here, IC chip fabrication was controlled with a blend of input-output control and myopic sequencing rules at each facility. No explicit attempt was made to match the production rate of any facility to the desired overall shop production rate. Lots were released to the floor as they were needed, as reflected in the backlog of customer orders for finished products. They were moved through the shop as the dispatchers noted inventory buildups and assigned jobs from those queues for immediate work. When lots were urgently needed, either as a consequence of changes in customer needs or as a result of manufacturing yield problems, they were expedited through the shop by giving them priority in each of the queues they entered.

The result was a large work-in-process inventory, at least compared to that which is achievable with detailed scheduling methods. The method described here appears to be capable of reducing the work-in-process inventory by a factor of two or more.

It should be pointed out that the company's objective under the former system had been one of maximizing worker utilization. To do that they wanted to make sure that material was always available for any worker to process. That objective has been modified somewhat and now they are prepared to accept some loss of labor efficiency (idle time) in exchange for reduced inprocess inventory and associated throughput time.

In the parlance of the job shop scheduling literature (e.g., Graves [2] or Baker [1]), this problem could be called a single job, n machine problem where there are m sequential tasks to perform to complete the job, with m is greater than n. Yet the problem is not to schedule the single job, but rather to schedule the manufacturing operation to continue to produce identical jobs most economically for an indefinitely long time. The key aspects of such a schedule are the throughput time for a single job and the overall production rate for the operation. The production rate dictates the utilization rate of the facilities and is constrained by the bottleneck facility or facilities. For a given production rate, the throughput time dictates the work-in-process inventory level. We describe the problem as one of attempting to reduce the throughput time per job as much as possible for a specified production rate.

We can view this problem in terms of the management of the facilities and, specifically, the sequence and timing of tasks to be performed at each facility. First, one must determine whether the desired production rate is feasible by measuring the rate at which each facility must work to provide the desired production rate. If the production rate is one job every ten minutes then each facility must be able to accomplish all the tasks required of it on one job within ten minutes. This might be one three-minute task, one two-minute task and three one-minute tasks. If there is enough time to accomplish all the tasks it will be possible to find a schedule, although that schedule may have an unacceptably long throughput time.

This is a hierarchical planning method [3] since a common production rate capability is first established at each facility and then the detailed sequence of activities at each facility is determined and performed at exactly that rate. When the production rate is changed the schedule must be revised. Both the sequence and detailed timing of tasks will change.

In the remainder of this paper we develop a method for preparing cyclic schedules for such an operation. The problem is described more precisely in the next section; an example of a cyclic schedule is developed in Section III. The details of the scheduling algorithm are developed in Section IV and results are given in Section V. In Section VI we discuss these results and the method.

II. Problem Statement

Many repetitions of the same job are to be completed. Each job consists of a specified sequence of tasks to be performed using a fixed set of facilities. The labor required to accomplish these tasks is fixed in amount and always available so that the performance of any task is not delayed because of labor unavailability. The cost of labor does not depend on the sequence of tasks chosen for any facility.

The production capacity or capability at any facility is fixed; it is established outside of and in advance of this scheduling activity in a hierarchical planning fashion.

The lot sizes or job sizes to be used are not a part of the problem. We assume the changeover costs from one job to another are zero at all facilities. In a flexible machining system the lot is one piece or the contents of one pallet. In integrated circuit chip fabrication the lot sizes are determined when designing the production machinery and are no longer variable. For example, furnaces are designed to hold racks with a specific capacity that becomes the lot size.

Each job may be processed more than once by each facility, i.e., a facility may perform more than one task on each job. The scheduling problem is that of finding the sequence and timing of tasks at each facility which will minimize the average throughput time for all the jobs, at a specified production rate. This is equivalent to minimizing the in-process inventory which is equal to the production rate times the throughput time.

A facility, as used here, may be either:

i) a single piece of equipment which can process one job (or lot) at a time, or

ii) multiple pieces of identical equipment, or a multiple capability single piece of equipment, which act as facilities in parallel and which can process more than one job (or lot) at a time, independently. We will refer to such facilities or facility centers as "multiple-channel" facilities, or

iii) a single piece of equipment which normally processes two or more jobs or lots simultaneously, in a coordinated way. That is, the facility performs a task on a "batch" of two or more jobs. We will refer to such a facility as a "batch facility." The number of jobs processed at one time is the "batch capacity."

Each channel of a multiple-channel facility could have a multiple-job batch capacity.

Several different kinds or classes of schedules are available for such a production system:

1. Myopic Job Sequencing

One approach to this problem is to devise rules for sequencing the jobs available for work at any facility. This is an approach that has been used for a large number of similar (job shop scheduling) problems in which jobs arrive at a facility from a number of predecessor facilities and then go on to one of several successor facilities. When there are a large number of tasks to perform and a large number of jobs in process at the same time it is very difficult to determine the best detailed flow or sequence of tasks through all the facilities. The myopic sequencing rules are used as a feasible alternative to detailed schedules.

2. Detailed Optimization

One can imagine an optimization procedure that examines all the possible feasible sequences of tasks at each facility which will complete a specified number of jobs within some specified time, i.e., obtain a required production rate. However, the number of feasible sequences is so large that this approach is thought to be computationally infeasible.

3. Cyclic Schedules

Since a specified production rate is to be achieved, jobs must be started at that average rate. Similarly, each task that is required to complete a job must be performed at that rate. That is, each facility must complete each of the tasks to be performed by it at that rate. A facility which accomplishes six different tasks on one job (and therefore on every job) must complete each of those at the overall production rate of the shop.

We can define a cycle with length equal to the reciprocal of the production rate. For example, a production rate of five lots every eight hour shift can be accomplished with a production cycle of one lot every 96 minutes. Each facility will have to complete each task required of it (to complete one job) once every 96 minutes.

On the average each facility has to do all the tasks required to complete a job once each cycle. If we schedule each facility to do each task _exactly once each cycle_, then we have a cyclic schedule that repeats each cycle. It is always feasible to generate a cyclic schedule. We just order the tasks in any sequence at the facility. It must be possible to accomplish all of them in one cycle, otherwise the overall production rate is not feasible. After each task the job waits until the next task, possibly in the next cycle. The schedule is feasible but may require a long throughput time.

This cyclic scheduling approach is the one we have used here. It reduces our problem to the scheduling of tasks during one cycle at each facility so as to minimize the throughput time of each lot.

III. Illustrative Example

A simple example will illustrate the concept of cyclic schedules. The tasks and the times required to complete a job (Job A) are shown in Gantt chart form in Figure 1. There are two tasks requiring a total of two hours to be done on Facility F1, two requiring six hours on F2, two (four hours) on F3, three (five hours) on F4, and two (four hours) on F5. If all these

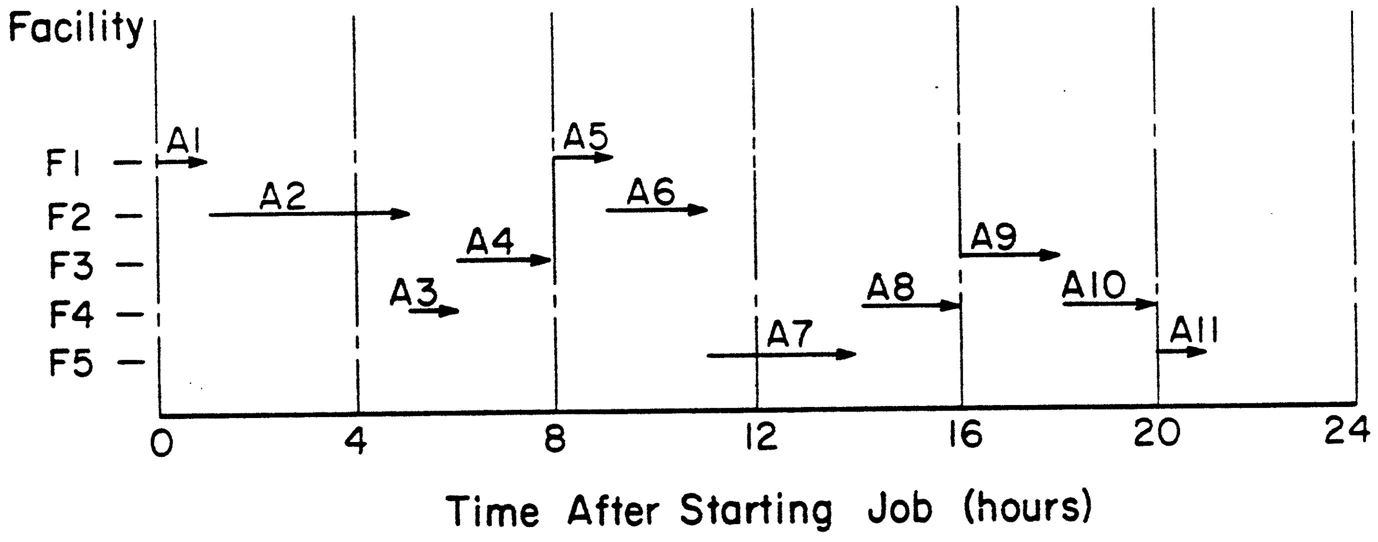

Figure 1. Gantt Chart Schedule for Job A

tasks were to be done in sequence with no delay the total throughput time would be 21 hours, the total production time required.

If we wanted a production rate of one lot per each eight hour shift (each job has the same process times and routing through the shop), we need to begin a second job (Job B) through a similar process starting eight hours later. That is not possible because Facility 1 is occupied at that time with Task 5 on the lot which started eight hours earlier.

Several possibilities come to mind to deal with this situation. One of these is to delay Task A5 for an hour in order to get the second lot started even though that delays the completion of Task A5 (and all subsequent tasks) for an hour. Also we find that we further need to delay Task A6 or we cannot complete Task B2 having started it immediately after Task B1 as we did with Task A2. So we delay starting Task A6 for three hours after completing Task A5 (Figure 2).

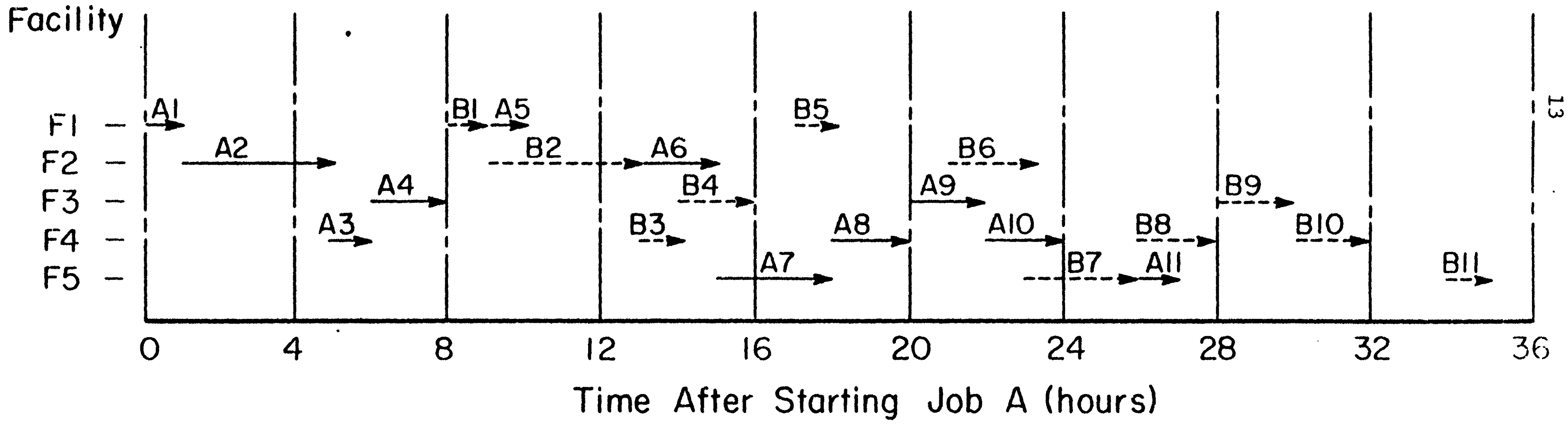

Figure 2. Gantt Chart Schedule for Two Jobs in Sequence
(8 hour interval)

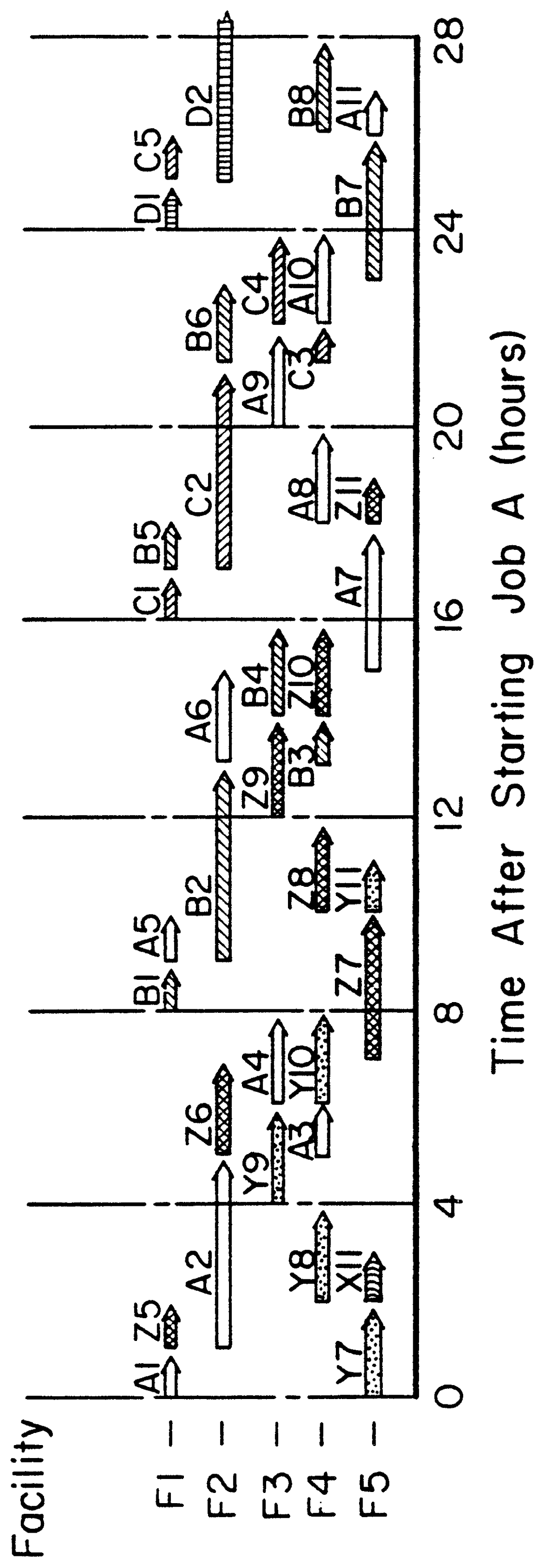

Figure 3. Section of Gantt Chart of Indefinite Duration

We then proceed through Tasks A7, A8, A9, and A10 with no further delay, although all of these are delayed four hours relative to the original schedule. At the end we have to delay beginning A11 for two hours, leading to a total delay of six hours in completing Job A relative to the 21 hours required if we made no provision for Job B. Note that now Job B has a schedule just like that of Job A although it starts eight hours later.

Having done this for Jobs A and B we now inquire whether a third job, Job C, could be started eight hours after Job B. It can, as shown in Figure 3, which also shows the earlier jobs, Jobs Z, Y and X, beginning 8, 16 and 24 hours before Job A.

The pattern in Figure 3 can be repeated for as long as we please. The pattern is cyclic with a job starting every eight hours.

Examining the first eight hour cycle we find that all the tasks required for a lot are performed there. Consequently, we can completely describe a schedule by considering the activities in one cycle (8 hours long in this case). We call this a Cyclic Gantt Chart and show this in Figure 4. It is very convenient to depict cyclic schedules in this way. In Figure 4, after

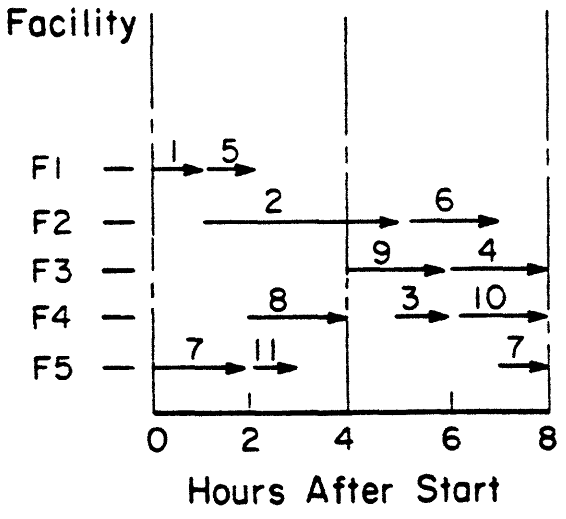

Figure 4. Cyclic Gantt Chart

completing Task 4, we "roll around" to the beginning of the cycle, find that Task 5 must be delayed for an hour and proceed. Task 6 must be delayed until the completion of Task 2 on the succeeding job (on Facility 2). Task 7 starts just before the end of the cycle and "wraps around" to the beginning of the next. Finally, although Task 10 completes at the end of the cycle, Task 11 must be delayed until Facility 5 is free, after completion of Task 7 on the following lot.

We note that each job requires three cycles plus three hours to complete, taking 27 hours instead of the 21 hours required for a single job with no interference.

There is no guarantee that a schedule prepared in this fashion obtains the minimum throughput time for a job.

In this example the cycle length is eight hours, corresponding to a production rate of one lot per shift. The most-heavily-loaded facility, F2, is occupied for only six of the eight hours in the cycle. Thus it seems that a considerably higher production rate is feasible even though we might expect the throughput time to increase. It is easy to construct schedules (for this small example) with various cycle lengths (production rates). As the cycle length is reduced to seven and six hours the throughput time first increases to 32 hours and then decreases to 29 hours (Figure 5), using the algorithm described in the next section. However, by exchanging the positions of Tasks 8 and 10 in the seven hour cycle of Figure 5, the overall job can be completed in 27 hours (three cycles plus six hours). Even

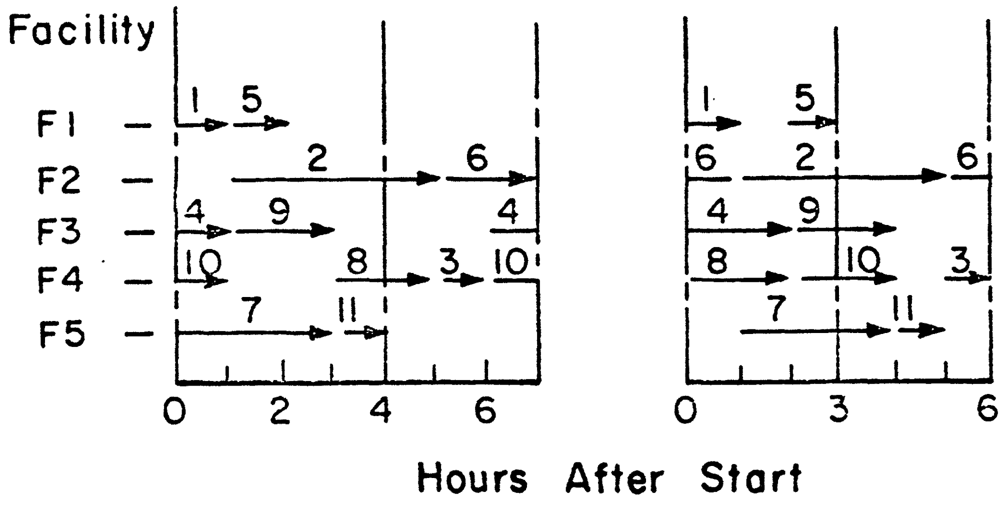

Figure 5. Example Schedules with Shorter Cycles

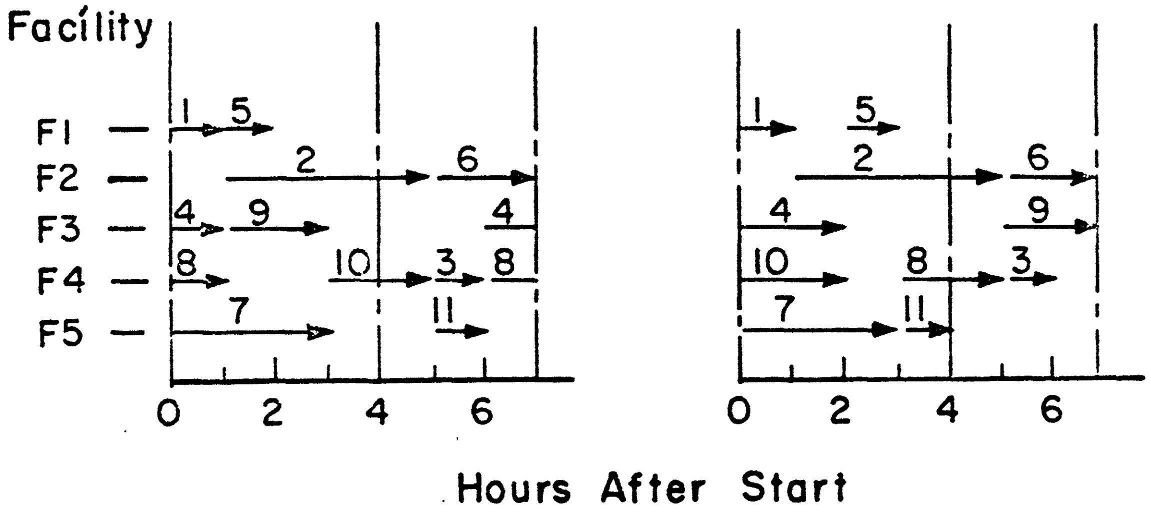

Figure 6. Alternative Seven Hour Cycle Schedules

further improvement is available by delaying the start of Task 4 for one hour, which does not add to the total delay in the system since Task 6 is already delayed for three hours. The delay in Task 4 allows Tasks 8, 9 and 10 to be done with no delay and the entire job is finished in 25 hours. These two schedules are shown in Figure 6.

As the production rate is decreased, the shop becomes less congested and it becomes easier to approach the minimum possible throughput time. Extending the cycle time from nine to fifteen hours in one hour increments yields the following job throughput times: 29, 25, 26, 21, 22, 23, 21. From the 15 hour schedule one can see that there will be no further interference as the production rate decreases. Of course, at this point facility utilization is very low.

While we would expect the throughput time to increase as the cycle time decreases, we see from the example that the increase need not be strictly monotonic. Rather, due to the combinatorial nature of the problem, decreasing the production rate may increase the interference among jobs.

The best schedule, the one with the minimum throughput time at a specified production rate, is the schedule with the right sequence and timing of tasks at each facility. This is a combinatorial optimization problem that we have not been able to solve. We have formulated it as an integer programming problem [5] but it does not appear to be computationally feasible even in this cyclic form.

In the special case where each facility performs only two tasks of equal duration and there are no multiple channel facilities and no facilities with batch (multiple lot) capability, the problem reduces to a network flow problem. We have not pursued that approach since that is much too restricted to fit the real case described in the Introduction. Because we could find no feasible optimization system for the problem, we resorted to a heuristic method similar to that described in the example problem.

IV. Heuristic Scheduling Algorithm

Basic Scheduling Procedure

The schedule is prepared as though one were preparing a Gantt chart for the cycle period. We first describe the algorithm when all facilities are single channel with a batch capacity of one job.

The first task is scheduled beginning at time zero, the second task begins at the time the first is complete. The third (or any later) task is scheduled on the appropriate facility, starting at the time the prior task is completed as long as there is no conflict with tasks already scheduled. .

Such a conflict may arise in either of two ways. A task already scheduled on the facility may not be complete at the time desired for commencement of the task to be scheduled, or there may be insufficient time available to complete the task between the desired starting time of the task to be scheduled and the commencement time of a task already scheduled. These conflicts are illustrated in Figure 7. Task j is completed on Facility F1 at Time 2. We would like to start Task j+1 on Facility 2 at Time 2 and occupy that facility until Time 4. If the facility is occupied by an earlier Task k less than j as indicated for Facility F2' we cannot start Task j+1 at Time 2 but must delay until Time 3. Similarly, even though the facility is not occupied at Time 2, as indicated for Facility F2" we cannot start at Time 2 since there is insufficient time to complete Task

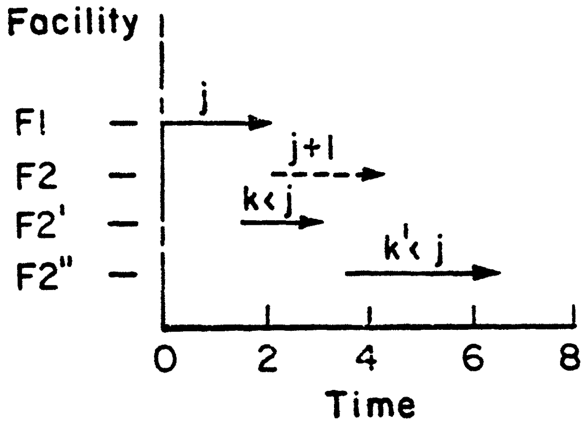

Figure 7. Illustration of Task Conflicts

j+1 before the facility is occupied by Task k' less than j. In either of these cases, commencement of the task to be scheduled is delayed until the conflict is removed. In this way, no previously scheduled tasks have to be rescheduled.

In scheduling these tasks, any task not complete at the end of the cycle is continued at the beginning of the cycle. That continuation represents, of course, the completion of the task commenced in the prior cycle.

There is one further very important restriction which we require in scheduling any task after the first one on any facility (or channel of a multiple channel facility). When the second or later task is scheduled on a facility, care must be taken to assure that the tasks not yet scheduled on that facility can all be accommodated in the remaining unscheduled blocks of time. This restriction guarantees that all the tasks to be accomplished by a facility can be scheduled without having to reschedule any tasks.

There are a number of ways to do this. We have adopted a very simple procedure. We require the schedule to leave a single uninterrupted block of time sufficient to accommodate <u>all</u> the tasks not yet scheduled on the facility.

The method is illustrated in Figure 8. Task j+1 is to be scheduled on Facility F2 on which Task k (with k less than j) has already been scheduled as shown. Task j+1 can start at Time 1 with no delay but that leaves two

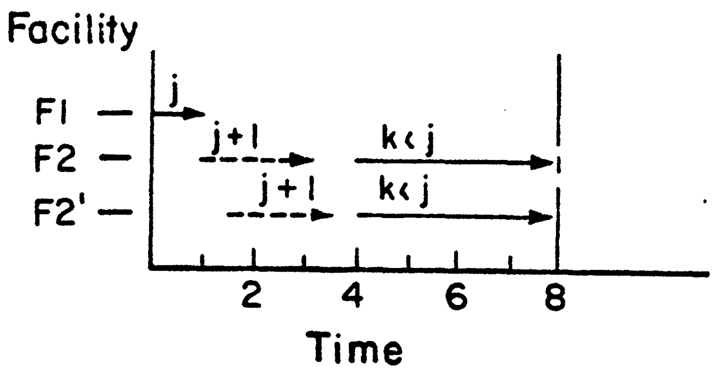

Figure 8. Procedure to Maintain Feasibility

blocks of time on Facility F2, each one hour long, neither sufficient to accommodate a third Task i (i is greater than j+1) which is one and one half hours long. To guarantee that space for Task i will be available, Task j+1 must be delayed for one half hour in starting.

If the number of tasks per facility is small this works quite well. If the number becomes large (perhaps greater than 5) it might be better to adopt a less restrictive method. An example of a less restrictive method is to require an uninterrupted block of time equal to or greater than some fixed fraction of the remaining unscheduled load. We have also experimented with a variable fraction with the fraction increasing to unity as the remaining tasks are scheduled. Both of these procedures reduce the delay introduced to guarantee feasibility.

Multiple Channel Facilities

In the case of multiple channel facilities the same basic procedure is used except for the feasibility guarantee. The task to be scheduled is scheduled on the channel which provides the least delay. If delay is equal at two or more channels, the task is arbitrarily scheduled on the first channel tried. To equalize the load among channels, tasks are preferentially scheduled on empty channels.

It is more difficult to guarantee feasibility of scheduling all the remaining unscheduled tasks. If the multiple-channel facility has substantial excess capacity, no problem arises; it will be possible to find sufficiently large time blocks to accommodate all the remaining tasks in a single channel of the facility. However, there is no simple way to guarantee feasibility at a multiple channel facility using this cyclic scheduling approach. Indeed, this problem is the combinatorial optimization problem known as the "bin packing" problem. When the requirement to leave sufficient time in a single block to process all remaining tasks on a single channel cannot be met, manual intervention has thus far allowed us to prepare a reasonably good schedule.

Batch Facilities

At a facility with multiple job batch capacity the first job arriving waits (before the needed task starts) until all the jobs needed in a batch have arrived. This means that it waits one cycle for each additional job in the batch.

At the completion of the batch process all the jobs but the first wait for processing of the next task to commence. Again, the wait for the last job in the batch is one less than the batch size times the cycle length.

The total processing time at a batch facility can be greater than the cycle length as long as the total processing time is less than the cycle length times the number of jobs in a batch. If this condition is met, the average facility tie up time per job is less than the cycle length.

Scheduling Algorithm Logic

Figure 9 is an overview flow chart of the computer program used to generate cyclic schedules. The program proceeds as follows:

Step 1. Feasibility Check

Check to make sure there is sufficient capacity at each facility to handle the desired production rate.

Step 2. Termination Step

The remaining steps proceed as a loop executed for each operation or task to be accomplished. Once a task is scheduled nothing further is done to it. When all tasks have been scheduled, the shop schedule is complete and the procedure stops.

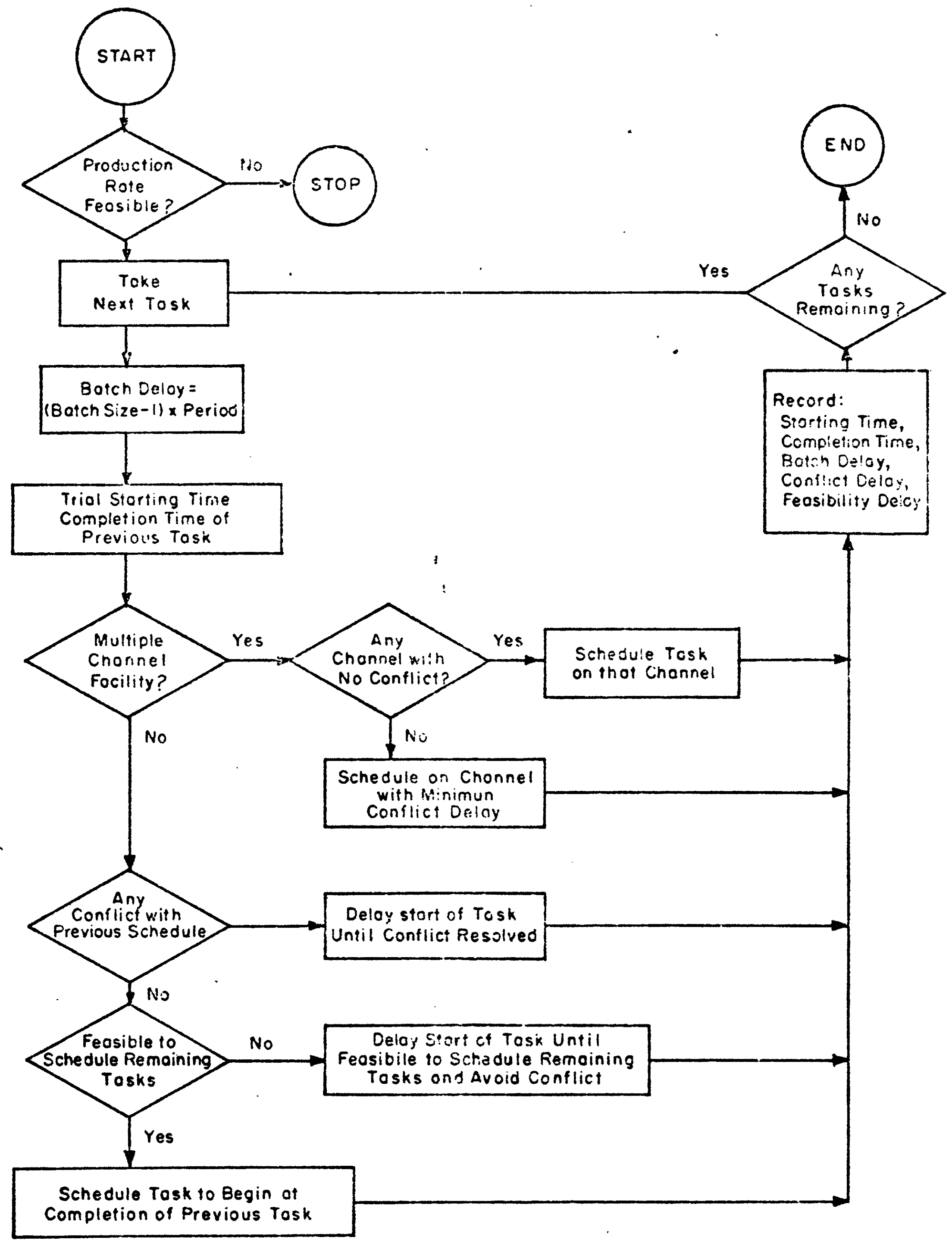
START
Production Rate Feasible ?
No
STOP
Take Next Task
Batch Delay = (Batch Size - 1) x Period
Trial Starting Time Completion Time of Previous Task
Multiple Channel Facility?
Yes
Any Channel with No Conflict?
Yes
Schedule Task on that Channel
No
Schedule on Channel with Minimun Conflict Delay
No
Any Conflict with Previous Schedule
Delay start of Task Until Conflict Resolved
No
Feasible to Schedule Remaining Tasks
No
Delay Start of Task Until Feasibile to Schedule Remaining Tasks and Avoid Conflict
Yes
Schedule Task to Begin at Completion of Previous Task
Record: Starting Time, Completion Time, Batch Delay, Conflict Delay, Feasibility Delay
Any Tasks Remaining ?
Yes
No
END

Figure 9. An Overview Flowchart of the Scheduling Logic

For each task, calculate the delay which results as a consequence of batching. Calculate also a trial starting time equal to the completion time of the previous task.

Step 3. Conflict Check and Multiple Channel Branch

If the facility has multiple channels, go to Step 5. Otherwise, proceed.

If the trial starting time leads to a task schedule which overlaps (conflicts with) any previous task schedule, delay the task starting time until no conflict exists. Go to Step 6.

If there is no conflict, go to Step 4.

Step 4. Feasibility Maintenance

Check to see whether a remaining single block of time is large enough to accommodate all the remaining unscheduled tasks. If so, the trial starting time is satisfactory. Go to Step 6.

If none of the remaining blocks of time is large enough, delay the task starting time until a satisfactory block remains. Go to Step 6.

Step 5. Multiple Channel

The scheduling of initial tasks on multiple channel facilities was omitted from the flow chart. Initially, tasks are scheduled sequentially on the channels until all channels have one task scheduled. Then the logic proceeds in a fashion similar to that for single channel facilities.

If a task can be scheduled with no delay, it is scheduled on the first channel encountered which provides no delay.

If no channel provides zero delay, the task is scheduled on the channel which provides the minimum delay.

Note that no feasibility guarantee is incorporated here. The single channel logic is inappropriate and was not needed in practice. The

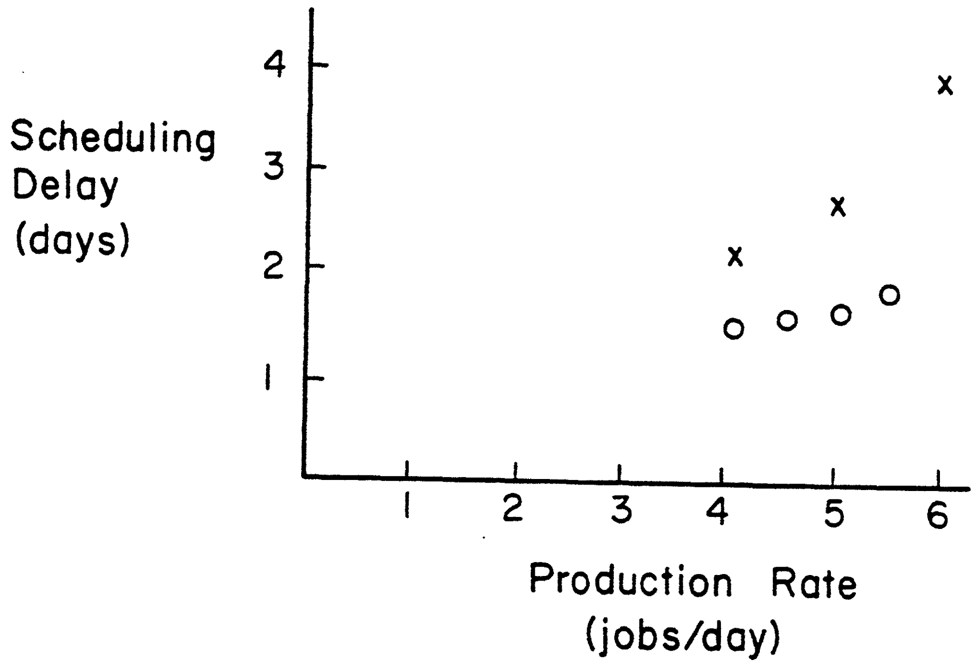

Figure 10. Results Comparison
o - Cyclic Schedules
x - Simulation

multiple channel facilities were either lightly loaded or had no more than two tasks per channel per cycle, so feasibility maintenance was not a problem.

Step 6. Recording

The schedule and associated delays are recorded. Return to Step 2.

V. Results

In the production process described in the Introduction there are 185 tasks for each job and 69 facilities. Of these facilities 13 are multiple channel and 3 have batch capacity greater than one. None of the multiple channel facilities has multiple job batch capacity.

The maximum capacity of the system is more than six jobs per day. As the production rate was increased from four to five and one-half jobs per day the throughput time per job varied as shown in Figure 10.

A deterministic simulation of the same system using first-in-first-out discipline at each facility yielded the simulation results also shown in Figure 10. The primary differences in performance seem to be in occasionally departing from the FIFO sequence in the cyclic schedule and in the handling of batch delays in the simulation. Since that schedule is not cyclic, all jobs can proceed immediately after batch completion.

This algorithm was developed in an attempt to improve the throughput times obtained with an earlier method, which was equally simple but gave poorer results. In that method we first found the earliest starting time for each task at each facility ignoring interference with other tasks. This corresponds to cutting Figure 1 into cycle length increments and superimposing them. Clearly, there will be conflicts at several facilities.

These conflicts were resolved one facility at a time by maintaining the starting sequence and delaying the start of each successive task just enough to resolve the conflicts. This method allows one to treat each facility independently and seems to be conceptually simpler than the system described above. However, ignoring the facility-to-facility transition in preparing a sequence for each facility leads to some long delays and obvious improvement possibilities. Many of those improvements were obtained by the algorithm described here which reduced the delays in the first schedules by about 40%.

VI. Comments

Hierarchical Character of the Planning Structure

The system described here is hierarchical in nature. The production rate (overall output rate of the shop) is selected first; the schedule of the tasks to be performed at each facility, to operate at the specified production rate, is then established to satisfy that constraint. As such, it is a <u>top-down</u> system; the production rate (primarily manpower level) is established first with a longer planning horizon than the schedule of tasks.

This system replaced a <u>bottom-up</u> system in which the detailed needs for specific lots were used to set priorities at each facility. As the load (backlog) at each facility grew it provided motivation to increase the production rate (workforce level) at that facility. Often a "bubble" of inventory would move through the shop and the workforce would be reassigned frequently to maintain this movement. That inventory bubble also increased the throughput time for all of the jobs.

By shifting the point of view, from bottom-up to top-down, as suggested by the hierarchical approach [4], a substantial reduction in throughput time has been obtained while also stabilizing the workforce assignments. The labor efficiency is decreased, however, since the schedule attempts to have the workforce wait for the jobs instead of having the jobs wait for the workforce.

Uncertainties

The procedure developed here makes no provision for uncertainties in machine or operator availability or variability in process time. In the process studied there was essentially no variability or uncertainty in process time. However, there is uncertainty in operator availability because of the conflicting demands on an operator's time from the multiple facilities attended by a single operator. Also there is uncertainty in facility availability as a result of machine breakdown.

In order to operate smoothly in the face of these potential interruptions of the flow, buffer times must be provided. These are located just downstream of facilities which are most subject to disruption because of breakdown or operator absence. Initially, we set these arbitrarily to cover the bulk of the down time distributions observed at these facilities. These buffers allow the remainder of the process to continue working according to schedule even though one facility is out of action. The stock represented by the buffer time maintains the flow until the time is exhausted, at which point downstream facilities run out of material and must also shut down.

After the facility is repaired action must be taken to restore the buffer time. If the facility has excess capacity available, this can be done during the regular work week. If not, overtime must be scheduled to recover the lost production time.

These buffers need to be kept small. Each time the facility performs a task the buffer time is added to the total throughput time of a job. Thus, if a facility performs four tasks and has a two hour buffer stock for each task, this buffer will increase the total throughput time by eight hours. We have used single-stage inventory theory to make an initial estimate of the required buffer stock size. This appears to overestimate the requirement and we are now attempting a multi-stage analysis to model the re-entrant characteristics of the flow process. Downstream facility down time reduces the demand on a buffer so the buffer need not be so large.

Queue Discipline

The schedule calls for a fixed sequence of tasks at each facility and, in turn, depends on having that sequence maintained. This means that the queue discipline cannot be described by any of the conventional myopic rules such as first-in-first-out, shortest-operating-time, etc. Instead, the processing priority is very idiosyncratic at each facility and is specified by the Cyclic Gantt Chart. The buffer stocks allow maintenance of this sequence in the presence of local interruptions of short duration. If there is a major breakdown the entire shop may be shut down.

Scrap

If an entire lot is scrapped part way through the process it is not available for further tasks and the time planned for those tasks will be idle, reducing efficiency. A job could be taken from a buffer and entered into the flow. We have made no attempt to address this problem in the scheduling system described here. The primary difficulty is the fact that the tasks at any one facility would have to be performed at different rates depending on whether they are early or late in the process, if some lots are not available for later tasks. This means that the cyclic procedure will not work because all the tasks at one facility do not have the same cycle length. This scrapping of a full lot is characteristic of a flexible manufacturing system in which the lot size is one piece. On the other hand, such systems operate at very low scrap rates so this may not be a significant problem. In integrated circuit wafer-processing the scrapping of a full lot is rare even though the number of wafers remaining in the lot declines as wafers individually are scrapped as the lot progresses. Since the lot continues to be processed as a unit the schedule is not modified.

Implementation Issues

The primary problem in the implementation of such a system is that of changing the approach to controlling the flow through the shop. A control process that has been drawing the next lot to process from a fairly large queue has to be changed to one of processing lots in strict sequence with very small queues.

There is no problem in staying on schedule as long as there are no breakdowns or operator unavailability. All the facility loads have been calculated and compared with capacity. All the operators have to do is follow the Gantt chart. The Gantt chart has to be translated into a shift schedule and the times measured from the beginning of a cycle have to be converted to clock time. It may not be necessary to identify the tasks with lot or job numbers on which the task is to be performed since there will generally be only one lot ready for the task called for. Another lot may be available in the queue or buffer but it will be waiting for some other task.

The shop needs to know what to do in the event of a breakdown. The shop operates with some delay relative to schedule until that breakdown is repaired and the shop is back on schedule. We expect that revised schedules will be issued for the operations affected.

Similarly, the shop needs to know how to get from these delayed schedules back to the regular schedule. This is mostly a question of operating so as to restore the buffers to their desired levels. Feasible schedules to do this will be devised. These involve running different tasks in the operation at different rates. This can be done on weekends or by changing the staff allocations to correspond to the desired production rate differences.

A similar problem exists when the production rate is changed. The sequences at the two rates may be different but it may be possible to change the sequence by using the buffers. The buffers in the system will have to be changed to correspond to the new rate. The recovery system just mentioned will have to be invoked to restore the desired buffer levels.

Expediting

With a schedule of this sort expediting is no longer possible, nor is it desirable. The schedule as developed should describe nearly the minimum feasible throughput time at the specified production rate. The schedule is facility limited rather than manpower limited, so adding labor will not speed things up. Changing the sequence (priority) of lots at a facility will not change the delay in completing a job.

Examination of the schedules developed for shop use shows that there is seldom a job waiting when a job arrives at a facility. (This can also be seen in Figure 3.) It appears, therefore, that the throughput time is close enough to the theoretical minimum that expediting is not desirable.

Multiple Products

The system described works when all the lots to be processed through the shop follow the same routing or process sheet. In integrated circuit wafer fabrication this is common when a number of different products differ only in the patterns placed on the wafers in the photolithographic steps. In flexible manufacturing systems this is characteristic of a family of parts differing only slightly in the placement or dimensions of holes or milling cuts.

When the products do not have identical process sheets, the applicability of the system depends on the degree of dissimilarity among products. If only the process _times_ required differ with different jobs, the longest time to do the corresponding task for any job can be used in the master process sheet to prepare the schedule and all jobs can be accommodated. This leads to some inefficiency since the jobs with the short processing times must wait during the remainder of the time provided for the longest processing time.

Even different routings can be fit into a common schedule if the differences are not too severe. If one product has a few extra operations the master process sheet can include these and the jobs which do not require them will wait during the time provided. Unfortunately the operators who have been made available to do those jobs will also wait, with resulting loss of labor efficiency. In some cases the reduction in inprocess inventory more than offsets this increase in labor cost. It is just a question of how similar the different processes are.

If the routings of two different families of products become very different but they still use the same facilities, one can develop an extended "process sheet" which looks like processing first one type of job and then the other. This is done with a cycle time long enough to produce one of each type and the jobs are released alternately. This restricts the output mix to be evenly divided between the two types of jobs. To provide a different mix, other cycles can be prepared, with different small number ratios of jobs of the two or more types. Then the desired output mix can be obtained over time by varying the fraction of the time each mix is produced. The cycle is longer but this does not affect throughput time a great deal. It is convenient to implement a schedule with a 24 hour cycle. Then each facility has the same schedule every day. However, this restricts the schedule to production rates which are integer numbers of jobs per day.

VII Conclusion

We have developed a cyclic method for scheduling re-entrant flow shops producing many jobs with similar or identical routings including multiple tasks at one or more facilities. These schedules are developed with a heuristic algorithm which may not be optimal but which appears to produce good schedules and is much more efficient than the usual myopic job sequencing rules characteristic of such shops. The system is deterministic but uses buffer times to protect the schedule from machine breakdown and operator unavailability effects. This method is being implemented in an integrated circuit chip fabrication plant and is expected to allow dramatic reductions in the inprocess inventory and job throughput time.

The scheduling algorithm itself is a small computer program with negligible running time. The schedule must then be translated into a shop schedule by accessing the current shop status and displaying to the operators the clock times at which various tasks should be performed on each lot. This latter process includes provision for breaks and lunch periods and modifications for weekend shut down and start up.

References

1. Baker, Kenneth R., Introduction to Scheduling and Sequencing, John Wiley and Sons, New York, 1974.

2. Graves, Stephen C., "A Review of Production Scheduling", Operations Research, 29 646-675, 1981.

3. Hax, Arnoldo C., and Harlan C. Meal, "Hierarchical Integration of Production Planning and Scheduling," in Logistics (M.A. Geisler, ed.) North Holland, Amsterdam, 1975.

4. Meal, Harlan C., Manfred H. Wachter and D. Clay Whybark, "MRP in Hierarchical Production Planning," EURO V-TIMS XXV, Lausanne, July 1982.

5. Stefek, Daniel, A Periodic Production Scheduling Problem, Master's Thesis, Massachusetts Institute of Technology, 1982.