

# MINIMIZING MAKESPAN WITH MULTIPLE ORDERS PER JOB IN A TWO MACHINE FLOWSHOP WITH ITEM PROCESSING

Jeffrey D. Laub, John W. Fowler, and Ahmet B. Keha  
*Arizona State University*

**Abstract:** New semiconductor wafer fabrication facilities use Front Opening Unified Pods (FOUPs) as a common unit of wafer transfer. Since the number of pods is limited due to high costs, and the increase in wafer size to 300-mm reduces the number of wafers required for a customer order, combining multiple orders into a single job is a necessity for efficient production. We investigate the multiple orders per job scheduling problem in the context of a 2-machine flowshop. A lower bound for the makespan performance measure is provided. An optimization model is presented that addresses both job formation and job sequencing.

**Key words:** flowshop, scheduling, multiple orders per job, optimization, semiconductor manufacturing, makespan, lot streaming

## 1. INTRODUCTION

The semiconductor manufacturing industry is one of the largest industries in the world, with annual revenues exceeding US \$200 Billion. With the extraordinary size of this market comes stiff competition and the need for manufacturers to continually improve their processes and equipment. One recent response to this challenge has been the increase of silicon wafer size from 200-mm to 300-mm. This results in approximately a 125% increase per wafer. However, since production lots typically contain 25 wafers, the lots are heavier than what can be physically carried safely between wafer processing tools. To address this, the industry has established a standard for the Front-Opening Unified Pod (FOUP) to transfer wafer lots between tools in a 300-mm wafer fab. These pods provide a clean atmosphere to prevent contamination of the wafer surface.

The increased wafer surface area results in a reduction in the number of wafers required to fill a customer's order. Assigning one FOUP to each order is a costly under-utilization of this scarce resource. Also, the processing time required by some operations is independent of the number of wafers. Sparsely filling FOUPs with individual wafers would adversely affect overall production efficiency. Semiconductor manufacturers using 300-mm wafers will need to group orders from different customers into a single FOUP, or job, which can then be routed to the appropriate processing tool. Mason, Qu, Kutanoğlu, and Fowler [1] and Kutanoğlu, Tanrisever, and Mason [2] and have addressed this problem domain, and refer to it as "multiple orders per job" (*moj*) scheduling problems.

Multiple orders per job problems are complex due to the interaction of two decisions that must be made, namely, job formation from orders and job scheduling. Further difficulties are imposed by the

restricted size of each job as well as the limited number of jobs (pods) available. Finally, there are at least two types of machines to consider for job processing. In *single lot processing*, such as wet sink processing, the processing time is independent of the number of items in the job. In *single item processing*, the processing time of a job is a function of the time required per item (wafer) as well as the number of items in the job. The effects of these machine properties is captured in the constraint field ( $\beta$ ) of scheduling notation as *moj(lot)* and *moj(item)* respectively. Note that the additional option of batching the formed jobs would be represented by *moj(●),batch*.

At this time, publications addressing the multiple orders per job problem deal with single machine configurations and either Total Completion Time or Total Weighted Completion Time performance measures. As semiconductor manufacturing processes typically entail a series of processes, the natural extension of analysis is the consideration of flowshops. This observation, plus the fact that the makespan is a typical performance measure of interest, motivates the proposed investigation of *Fm | moj(●) | Cmax* and related problems. The results of this paper establish a lower bound on the makespan for multiple orders per job problems in a 2-machine flowshop.

## 2. PROBLEM DEFINITION

We assume that a set of orders  $O=\{o_k\}$ ,  $k=1..H$  are available at time zero for processing and scheduling on  $m$  sequential machines. Each order has a size  $s_k$ , representing the number of items (wafers) in the order. In general, the processing time of a single item is dependent on its order and the machine performing the processing. In our problem, we restrict the item processing time to be independent of the order. Without loss of generality, each item requires one unit of processing time on the slowest machine. The processing time of each item is the same on any given machine, but the machine speeds may be different.

Each order must be assigned to exactly one of  $F$  jobs. Each job has a size capacity of  $K$ , and multiple orders can be assigned to a single job as long as the sum of order sizes does not exceed  $K$ , and no order is divided among multiple jobs (In the case where the size of an order exceeds  $K$ , we assume the order is first divided into separate orders, each of which is of size  $K$ , except for possibly one). No preemption of jobs or orders is allowed.

As discussed earlier, the processing time of a job depends on the type of machine. In an item-processing machine, denoted *moj(item)*, the job processing time is the sum of the orders' individual item processing times for that machine. The task is to assign the orders to jobs and schedule the jobs to minimize the makespan of the entire set of orders through a 2-machine flowshop.

To accommodate the problem description above, we use the following notation.

- $m$  : number of machines
- $M_i$  : machine  $i$ ,  $i \in \{1..m\}$
- $K$  : job capacity
- $F$  : maximum number of jobs allowed (FOUPs)
- $H$  : number of orders
- $o_k$  : order  $k$ ,  $k \in \{1..H\}$
- $\rho_{ik}$  : processing time for order  $k$  on machine  $i$ .
- $\tau_{ik}$  : processing time per item in order  $k$  on machine  $i$ .
- $s_k$  : size of order  $k$ . We assume  $s_k \leq K$ .
- $p_{i,j}$  : processing time for job  $j$  on machine  $i$ .  
 $= \sum \rho_{i,k}$  for  $o_k \in \text{Job } j$  for *moj(item)* machines

### 3. LITERATURE REVIEW

As the *moj* problem set is relatively new, there are few papers published on this subject. Mason, *et al.* [1] authored one of the most recent articles, and establishes much of the background motivation for the problem domain. Their paper provides a linear integer program for the single machine *moj*(•) Total Weighted Completion Time problems. Due to the lengthy execution time of this model, they propose and analyze combinations of 5 order sequencing rules, 2 job filling directions to select the orders from the sequence, and 6 order-to-job assignment rules specifying the job construction stage. Their results identify which heuristic combinations provide the best performance measurement while adhering to short execution times, dependent on problem instance characteristics.

Another recent paper by Kutanoglu, *et al.* [2] addresses the single machine *moj*(item) Total Completion Time problem. For the uncapacitated version, they devise an improvement-based algorithm and a lot-size-based algorithm for solving the problem to optimality. As the complexity of the former is exponential in the number of orders, it becomes ineffective at approximately 35 jobs. The latter is polynomial in the number of orders. The capacitated version is sufficiently complex to abandon attempts to achieve an optimal algorithm, and the authors provide a heuristic based on the prior optimal solutions by removing the capacity infeasibilities while preserving as much of the original schedule as possible. Their experiments demonstrate the effects of various problem parameters on the performance of the heuristic.

Prior to these contributions, another paper addressing a similar problem was Dobson and Nambimadom's [3] article on batch loading and scheduling. They address the mean weighted flow time objective for a single machine. However, the batch processing times depend only on the family, not the number or volume of jobs in a batch, and there is no restriction on the number of batches constructed.

We will demonstrate that by removing several of the restrictions of the multiple orders per job problem, its relaxation resembles a form of lot streaming. See Chang and Chiu [4] for a comprehensive survey of lot streaming results.

To the authors' knowledge, this is the first paper to address the multiple orders per job problem domain in the context of flowshops.

### 4. ITEM PROCESSING

For item-processing, we demonstrate that we only need to consider solutions that use all the available jobs ( $F$ ). This result is not restricted to the 2-machine case.

**Theorem 1.** For the  $F_m \mid \text{moj}(\text{item}), \text{prmu} \mid C_{\max}$  problem, there is an optimal schedule that uses  $\min(F, H)$  jobs.

**Proof.** Suppose there is no optimal schedule that uses  $\min(F, H)$  jobs. Then there is an optimal schedule  $S$  in which at least one job  $j$  contains more than one order, and there are  $n$  jobs where  $n < \min(F, H)$  (see Figure 1). Construct  $S'$  from  $S$  by removing one order from this job (creating job  $j'$ ) and creating a new job  $j''$  containing just the one order. Since  $p_{ij} = p_{ij'} + p_{ij''}$ , the original job  $j$  in  $S$  can be replaced by jobs  $j'$  and  $k$  in  $S'$  without increasing  $C_{\max}(S')$  (see Figure 2). This can be repeated until  $n$  is at least  $\min(F, H)$ . Thus the original assumption is invalid.

<<Figure 1>>

<<Figure 2>>

Since the case for  $H \leq F$  is solved by placing each order in a separate job, we will assume that  $H > F$ . We also assume that the processing time of all items is the same on a given machine ( $\tau_{ik} = \tau_i$ ). Consider the following instance:

- $\tau_{1,k} = 1$       All items require 1 time unit on machine 1
- $\tau_{2,k} = 2$       All items require 2 time units on machine 2
- $s_1 = 3$         The size of order 1 is 3 items
- $s_2 = 4$         The size of order 2 is 4 items
- $s_3 = 5$         The size of order 3 is 5 items
- $J_1 = \{o_1, o_2\}$     Job 1 consists of orders 1 and 2
- $J_2 = \{o_3\}$       Job 2 consists of order 3

Then the job processing times on each machine are as follows:

<<Table 1>>

Essentially, the machines have different rates, but the same machine rate applies to all items. This will result in jobs whose processing times equal the number of items in the job times the relative machine factor. Therefore, once the job composition has been determined, the sequencing problem becomes  $F2 | \text{prmu}, p_{i,j} = p_j/v_i | C_{\max}$ , where  $v_i$  is the relative machine speed. Applying this notation to the example above, we have  $p_1 = 7, p_2 = 5, v_1 = 1, v_2 = 1/2$ . This leads to the following result for the uncapacitated problem.

Lemma 1. The  $F2 | \text{moj}(\text{item}), \text{prmu}, \tau_{ik} = \tau_i | C_{\max}$  problem (with  $K = \infty$ ) is equivalent to finding the best assignment of orders to jobs for which the  $F2 | \text{prmu}, p_{i,j} = p_j/v_i | C_{\max}$  problem solution has the best  $C_{\max}$ .

## 5. OPTIMAL UNCAPACITATED JOB SIZES

The flexibility in these moj problems is the ability to somewhat control the job processing times via the chosen job composition (order assignment). For the 2-machine case, since we know that the SPT-LPT algorithm of Johnson [5] is optimal, the goal is to select the job composition for which the optimal solution is better than the optimal solutions for all other job compositions. Once job composition is determined, then optimal job sequencing is easily determined. In general, the task is to characterize the properties of a job mix that best exploits the optimal solutions to the problems. To find the optimal job sizes, we first consider the problem as if orders were infinitely divisible, and the total order processing time could be distributed arbitrarily among the jobs.

Since there is no idle time for  $M_1$ , and the sequence of job completion is irrelevant, the goal is to minimize the idle time on  $M_2$ . Intuitively, we'd like to get the first job finished on  $M_1$  quickly so that  $M_2$  can start work, and have no other idle time for  $M_2$ . Assuming in the first case that  $M_2$  is slower than  $M_1$ , we let the  $M_1$  speed,  $v_1$ , equal 1 and let  $c = 1/v_2$ , so that  $c > 1$ . Once job 1 has completed processing on  $M_1$ , since we know it will take  $cp_{1,1}$  time on  $M_2$ , the best choice for  $p_{1,2}$  is  $cp_{1,1}$ . In general, we would like  $p_{1,k+1} \leq p_{2,k}$ . This results in no idle time on  $M_2$  other than the wait for job 1 to finish on  $M_1$ . So we would then like to have as small a value for  $p_{1,1}$  as possible. This is depicted in the following diagram.

<<Figure 3>>

If we let  $P_1$  equal the total processing time requirement for all jobs on  $M_1$ , then the recursive relationship results in the following.

$$p_{1,k} = c^{k-1} p_{1,1} \text{ for } 2 \leq k \leq n$$

$$\sum_{j=1}^n p_{1,j} = P_1 = p_{1,1} \sum_{j=0}^{n-1} c^j$$

Thus, the optimal selection of processing times is geometric. The result, analogous to the result of Potts and Baker [6] regarding lot streaming, is as follows.

**Lemma 2:** For a given number of jobs  $n$ , and a given total  $P_1$  of processing times of the  $n$  jobs on machine 1, and a relative machine 2 rate of  $v_2 < 1$ , the geometric distribution (using factor  $c=1/v_2$ ) of this total produces the best  $C_{max}$  possible for  $F2 | pmu, p_{i,j} = p_j/v_i | C_{max}$ , among all partitions of  $P_1$  over  $n$ , i.e.,

$$p_{1,j} = P_1 \frac{c^{j-1}(c-1)}{c^n - 1} \quad (1)$$

**Lemma 3:** The optimal  $C_{max}$  is

$$P_1 \left( \frac{c^{n+1} - 1}{c^n - 1} \right) \quad (2)$$

A similar result holds when  $v_2 > 1$ , with jobs sequenced by decreasing processing time.

## 6. OPTIMAL CAPACITATED JOB SIZES

To apply this result to the *moj(item)* problem, suppose we have the sum of order processing times = 15 on machine  $M_1$ , and  $M_2$  runs 2 times as slow as  $M_1$ , and the maximum number of jobs (FOUPs) we can construct is 4. Then we want to allocate the orders to jobs (if possible) so that

$$p_{1,j} = P_1 \frac{c^{j-1}(c-1)}{c^n - 1} = (15) \frac{2^{j-1}}{15} = 2^{j-1}$$

The partitioning of  $P_1$  that meets this requirement is (1, 2, 4, 8), with  $C_{max} = 31$ . However, this does not consider the possible restriction imposed by the limiting job capacity,  $K$ . If  $K = 6$ , then the optimum partition is infeasible. If job 4 is reduced to the maximum processing time, the remaining jobs become a subproblem, with  $n'=3$ , and  $P' = 9$ . This produces a distribution of (1.29, 2.57, 5.14, 6.00) for the complete problem, with a  $C_{max}$  of 31.29.

Based on this strategy, the following theorem determines the optimal job sizes  $s$  for a given flowshop with machine 2 time factor equal to  $c$ , a total number of items  $P_1$ , a maximum number of jobs  $n$ , and a job capacity limit  $K$ .

**Theorem 2:** For  $F2|moj(item)|C_{max}$  problem, the following equations determine the optimal job processing times  $p_{i,j}$  for a given flowshop with machine 2 time factor  $c > 1$ , a total machine 1 processing time  $P_1$ , a maximum number of jobs  $n$ , and a job capacity limit  $K$ , where job size is equal to the job processing time.

If

$$b = \max \left\{ x \mid (P_1 - xK) \left( \frac{c^{n-1-x}(c-1)}{c^{n-x} - 1} \right) > K \right\}$$

for integer(x), ( $b$  is the number of capacitated jobs) then the optimal job processing times satisfy:

$$p_{1,j} = (P_1 - bK) \frac{c^{j-1}(c-1)}{c^{n-b} - 1} \quad 1 \leq j \leq n-b \quad (3)$$

$$p_{1,j} = K \quad n-b < j \leq n \quad (4)$$

Jobs with these processing times result in the following Cmax.

$$C_{\max} = (P_1 - bK) \left( \frac{c^{n-b+1} - 1}{c^{n-b} - 1} \right) + cbK \quad (5)$$

The structure of the proof is as follows. First, we show that for the processing times in equations (3) and (4), equation (5) specifies the corresponding value of Cmax. Next, we show the case for  $b=1$ , i.e., that the equations hold when only one job needs to be reduced in size from the geometric optimum to size  $K$ . We then use this fact as the basis of induction on  $b$ . We then show that the  $p_{i,j}$  defined by equations (3) and (4) sum to  $P$ .

Step 1. We show that equation (5) holds.

The Cmax for the job sequence defined by equations (3) and (4) is equal to the Cmax for the uncapacitated jobs plus the total processing time for the remaining jobs on machine 2, which takes  $cbK$  time. From Lemma 3, the Cmax for the uncapacitated jobs is

$$(P_1 - bK) \left( \frac{c^{n-b+1} - 1}{c^{n-b} - 1} \right)$$

Step 2. We prove that if  $P_1$  is large enough to require capacitation of exactly one job, then the hypothesis of Theorem 2 is true. This is captured by the following lemma.

Lemma 4: Let

$$(P_1 - K) \left( \frac{c^{n-2}(c-1)}{c^{n-1} - 1} \right) > K$$

$$(P_1 - 2K) \left( \frac{c^{n-3}(c-1)}{c^{n-2} - 1} \right) \leq K$$

Then equations (3) and (4) define the optimal processing times for the Cmax objective.

Proof (by contradiction): Suppose this is not the case. Then there is a superior set of  $p_{1,j}$  where either (1)  $p_{1,n} < K$ , or (2) the  $p_{1,j < n}$  are not geometric per equation (3).

Case 1: Suppose  $p_{1,n} < K$ .

Let  $p_{1,n} = K - \varepsilon$ . Let  $C_{\max}'$  be the optimal makespan for the first  $n-1$  jobs. From Lemma 3, we have

$$C_{\max}' = (P_1 - p_{1,n}) \left( \frac{c^n - 1}{c^{n-1} - 1} \right)$$

So the makespan for the entire job set is

$$C_{\max} = (P_1 - p_{1,n}) \left( \frac{c^n - 1}{c^{n-1} - 1} \right) + cp_{1,n}$$

Thus,

$$C_{\max} = (P_1 - K) \left( \frac{c^n - 1}{c^{n-1} - 1} \right) + cK + \left( \frac{c^n - 1}{c^{n-1} - 1} - c \right) \varepsilon$$

But this is larger than the  $C_{\max}$  of equation (5).

Case 2: Suppose that the first  $n-1$  job processing times are not geometric per equation (3).

The processing time distributed among these jobs is  $P_1 - K$ . From Lemma 2, the optimal distribution is

$$p_{1,j} = (P_1 - K) \frac{c^{j-1}(c-1)}{c^{n-1} - 1}$$

This is exactly equation (3) for  $b=1$ , thus contradictory.

End of Lemma 4 Proof

Step 3: We have shown that the theorem is true for the  $b=1$  case. We now complete the proof by induction on  $b$ . We will demonstrate that if the theorem holds for any  $b < q$ , then it holds for  $b = q$ .

Suppose  $b = q$ . Then

$$(P_1 - qK) \frac{c^{n-q-1}(c-1)}{c^{n-q} - 1} > K \quad (6)$$

$$(P_1 - (q+1)K) \frac{c^{n-q-2}(c-1)}{c^{n-q-1} - 1} \leq K \quad (7)$$

We will now show that the optimal distribution must have at least  $q$  jobs of processing time  $K$ . Assume there are only  $q-1$  such jobs. Then consider the remaining  $n' = n - q + 1$  jobs. The total processing time for this set is  $P' = P_1 - (q-1)K$ . We show that for this set of jobs,  $b' = 1$ . From equations (6) and (7) we have

$$(P' - K) \frac{c^{n'-2}(c-1)}{c^{n'-1} - 1} = (P_1 - qK) \frac{c^{n-q-1}(c-1)}{c^{n-q} - 1} > K$$

$$(P'-2K) \frac{c^{n'-3}(c-1)}{c^{n'-2}-1} = (P_1 - (q+1)K) \frac{c^{n-q-2}(c-1)}{c^{n-q-1}-1} \leq K$$

Therefore,  $b'=1$ .

This satisfies the induction hypothesis for  $b = q - 1$ . So the optimal distribution for the reduced job set is defined by equations (3) and (4) for  $P'$ ,  $m'$ , and  $n'$ . Looking at the last job in this sequence, we have

$$p_{1,n'} = p_{1,n-(q-1)} = (P_1 - qK) \frac{c^{n-q}(c-1)}{c^{n-q+1}-1} > (P_1 - qK) \frac{c^{n-q-1}(c-1)}{c^{n-q}} > K$$

This violates the assumption that there were less than  $q$  jobs of size  $K$  or greater. Thus, there must be at least  $q$  jobs of size  $K$ .

To show that there are no more than  $q$  jobs of size  $K$ , we suppose that there are  $q+1$  jobs of size  $K$  and demonstrate a contradiction. Consider the job set obtained by removing  $q$  jobs of processing time  $K$ . The last job in this remaining sequence has processing time  $K$ , and the total processing time is  $P'=P-qK$ . By Lemma 2, the last job should have processing time

$$p_{1,n-q} = (P_1 - qK) \frac{c^{n-q-1}(c-1)}{c^{n-q}-1}$$

But by equation (6), this is larger than  $K$ , which is a contradiction. Therefore, the original job set must have exactly  $q$  jobs with size  $K$ .

With  $q$  jobs of size  $K$ , the remaining processing time is  $P'=P-qK$ . Applying Lemma 2, equation (3) is met for  $b=q$ , and the induction is completed.

Step 4. We now show that  $\sum_{j=1}^n p_{1,j} = P_1$

$$\sum_{j=1}^n p_{i,j} = \sum_{j=1}^{n-b-1} p_{i,j} + bK = (P_1 - bK) \frac{c-1}{c^{n-b}-1} \sum_{j=0}^{n-b-1} c^j + bK = (P_1 - bK) \frac{c-1}{c^{n-b}-1} \frac{c^{n-b}-1}{c-1} + bK = P_1$$

*End of Theorem.*

A similar result holds when  $c < 1$ .

## 7. MATHEMATICAL FORMULATION

The following Mixed Integer Programming model represents the  $F_m \mid \text{moj}(\text{item}), \text{prmu}, \tau_{ik} = \tau_i \mid C_{\max}$  problem.

### Decision Variables:

$x_{ij}$  = start time of job  $j$  on machine  $i$ , for  $i \in \{1..m\}, j \in \{1..F\}$

$y_{jj'}$  = 1 if job  $j$  runs before job  $j'$ , 0 otherwise, for  $j \in \{1..F\}, j' \in \{j+1..F\}$

$z_{kj}$  = 1 if order  $k$  is assigned to job  $j$ , 0 otherwise, for  $k \in \{1..H\}, j \in \{1..F\}$ .

$p_{i,j}$  = processing time of job  $j$  on machine  $i$ .

Constants:

$m$  = number of machines

$s_k$  = number of items in order  $k$ , for  $k \in \{1..H\}$ .

$\tau_i$  = processing time for a single item on machine  $i$ , for  $i \in \{1..m\}$ ,  $\tau_1=1$ .

$K$  = maximum size of each job, i.e., maximum sum of the sizes of orders within a job.

$F$  = maximum number of jobs allowed

$M$  = very large number, e.g.,  $KF \sum_{i=1}^m \tau_i$ .

Objective Function: Minimize  $C_{max}$

Constraints:

[1] The processing time of a job is equal to the size of the job (sum of order sizes) times the item processing time.

$$p_{i,j} = \tau_i \sum_{k=1}^H s_k z_{k,j} \text{ for } i \in \{1..m\}, j \in \{1..F\}$$

[2] The objective function is the maximum of all job completion times.

$$C_{max} \geq x_{m,j} + p_{m,j} \quad \text{for } j \in \{1..F\}$$

[3] No two jobs  $j$  and  $j'$  can be scheduled on the same machine at the same time

$$\begin{aligned} x_{i,j} + p_{i,j} &\leq x_{i,j'} + M(1-y_{j,j'}) && \text{for } i \in \{1..m\}, j \in \{1..F\}, j' \in \{j+1..F\} \\ x_{i,j'} + p_{i,j'} &\leq x_{i,j} + My_{j,j'} && \text{for } i \in \{1..m\}, j \in \{1..F\}, j' \in \{j+1..F\} \end{aligned}$$

[4] A job must complete on machine  $i$  before it begins on machine  $i+1$ .

$$x_{i,j} + p_{i,j} \leq x_{i+1,j} \quad \text{for } i \in \{1..m-1\}, j \in \{1..F\}$$

[5] Each order is assigned to exactly one job.

$$\sum_{j=1}^F z_{k,j} = 1 \text{ for } k \in \{1..H\}$$

[6] No job size exceeds  $K$ .

$$\sum_{k=1}^H s_k z_{k,j} \leq K \text{ for } j \in \{1..F\}$$

Sample problem instances were prepared using the parameters of the experiment design described in Mason, *et al.* [1]. Using the 15-order case ( $H=15$ ), the levels for the number of jobs  $F$ , the job capacity  $K$ , and the distribution of order sizes are determined by the following table, where  $v \in \{3,5\}$  and  $B \in \{1,2\}$ .

<<Table 2>>

Ten instances for each of the four combinations of  $v$  and  $B$  were generated, with the following results. The optimal makespan is compared to the lower bound from Theorem 2. The execution times are in seconds on a 2.4GHz personal computer.

<<Table 3>>

A small increase in the number of available jobs ( $F$ ) from 7 to 8 in Test Set #2 resulted in execution times that were approximately 30 to 60 times longer. In some cases, the runs were terminated after hours of processing without making progress on the best solution found.

## 8. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced the flowshop configuration of the multiple orders per job problem domain. This new problem domain represents the challenges faced by semiconductor manufacturing plants where a fixed quantity of expensive transport equipment must be used effectively to keep production costs low and deliveries on time. The problem requires aggregating orders into jobs, and sequencing those jobs in a flowshop under the restrictions of a limited job capacity, a fixed number of jobs, and non-identical order sizes. Equations are provided for the optimal mix of job sizes to establish a lower bound for the makespan performance criterion. A mixed integer programming model is defined that accounts for the optimal assignment of orders to jobs as well as job sequencing.

The properties of the multiple orders per job problem, especially in a flowshop configuration, provides for a rich problem set. The compute time for calculating optimal solutions for these complex problems is excessive even for relatively small problem instances. The authors are currently investigating using the lower bound in a branch and bound algorithm, and are developing heuristics for various configurations of flowshops and performance measures for the multiple orders per job problem set. We expect that these heuristics will leverage prior work in single-machine counterparts, possibly in conjunction with metaheuristics such as genetic algorithms and Tabu search.

## 9. BIBLIOGRAPHY

- [1] S. J. Mason, P. Qu, E. Kutanoglu, and J. W. Fowler, "The single machine multiple orders per job scheduling problem", ASU Working Paper ASUIE-ORPS-2004-004, 2004.
- [2] F. Tanrisever, E. Kutanoglu, and S.J. Mason, "Forming and Scheduling Jobs from Multiple Orders with Different Attributes: A Computational Study of Special Cases," Proceedings of 13th Industrial Engineering Research Conference, IIE, Houston, TX, May, 2004.
- [3] G. Dobson and R.S. Nambimadom, The batch loading and scheduling problem, *Operations Research*, 2001, Vol. 49, No. 1, 52-65.
- [4] Jen Huei Chang and Huan Neng Chiu, A comprehensive review of lot streaming, *International Journal of Production Research*, 2005, Vol. 43, No. 8, 1515-1536.
- [5] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, 1954, Vol. 1, pp 61-68.
- [6] C. N. Potts and K. R. Baker, Flow Shop Scheduling with Lot Streaming, *Operations Research Letters*, 1989, Vol. 8, pp 297-303.

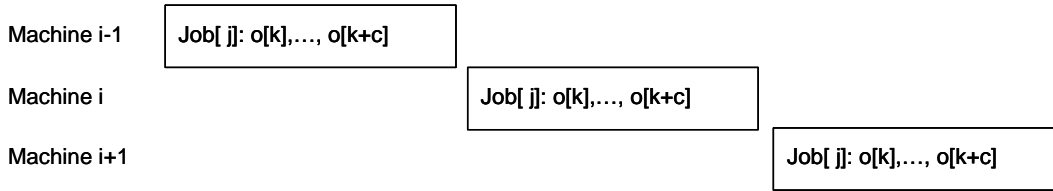


Figure 1. Schedule S with n jobs,  $n < \min(F,H)$

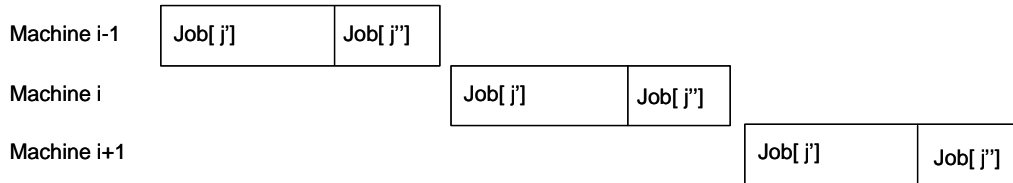


Figure 2. Schedule S' with n+1 jobs

Table 1. Job Processing Times

	J1	J2
M1	7	5
M2	14	10

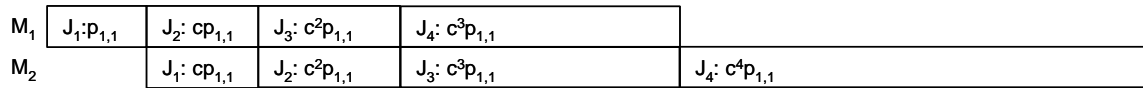


Figure 3. Optimal Job Sizes

Table 2. Experiment Parameters

Order Sizes	$DU[v-(v+1)/2, v+(v+1)/2]$
Capacity (K)	12B+1
Jobs (F)	$\text{ceiling}(Hv/12B)+1$

Table 3. Experiment Results

Test Set	K	F	H	Order Range	Avg Cmax/LB	Avg Time
1	13	5	15	1..5	1.018	<1
2	13	7	15	2..8	1.021	141
3	25	3	15	1..5	1.004	<1
4	25	4	15	2..8	1.019	<1