

# MULTIPLE ORDERS PER JOB BATCH SCHEDULING WITH INCOMPATIBLE JOBS

Vishnu Erramilli, Scott J. Mason

*Department of Industrial Engineering, 4207 Bell Engineering Center, University of Arkansas, Fayetteville, AR 72701, USA*

**Abstract:** The growth in demand for semiconductors has been accompanied by intense competition between semiconductor manufacturers to satisfy customer on-time delivery needs. This paper is motivated by scheduling the wafer fabrication phase of 300-mm semiconductor manufacturing, as it contains both batch processing operations and multiple customer orders being grouped into production jobs. The resulting *multiple orders per job (MOJ)* scheduling problem is presented for the batch-processing environment as exemplified by a diffusion oven. A mixed-integer programming formulation is presented for incompatible job families wherein only jobs that belong to the same family may be grouped together in a batch. This formulation is analyzed using an experimental design to minimize total weighted tardiness. While optimal solutions are attainable for small problem instances in a reasonable amount of computation time, heuristic development is warranted to support the analysis of larger, more practical MOJ batch scheduling problems. Promising heuristic approaches are identified that find near-optimal solutions very quickly.

**Key words:** semiconductor manufacturing, batching, total weighted tardiness

## 1. INTRODUCTION

The semiconductor manufacturing industry is one of the major industries in the world. Tremendous industrial growth has taken place over the last decade due to advancements in production technology and operational principles. The growth in demand for semiconductor products has been accompanied by intense competition between semiconductor manufacturers to satisfy customer needs. Satisfaction of customer needs is a very important part of any business. A very important dimension of customer satisfaction and goodwill is the on-time delivery of products. Indeed, the goal of most manufacturers is on-time delivery of high quality products while simultaneously reducing production costs.

On-time delivery of products can be promoted by reducing production cycle times. Cycle time reduction can be realized by purchasing and using extra equipment in the factory. However, tooling costs in the semiconductor manufacturing industry are very high, approaching \$15 million per tool; therefore, this option often is ruled out due to financial reasons. Alternately, production cycle times may be reduced more cost-effectively by efficiently planning and scheduling the existing resources in a semiconductor manufacturing facility (“wafer fab”). Planning and scheduling can lead to efficient utilization of production resources, which in turn can lower the time required to produce a product, promoting on-time delivery.

The typical semiconductor manufacturing process can be divided into four sub-systems/stages: wafer fabrication, wafer probe, wafer assembly, and wafer (final) testing (Uzsoy *et al.* 1992). Wafer fabrication and wafer probe are normally referred to as the “front-end” of the production process, while wafer assembly and final testing are commonly termed the “back-end.” In wafer fabrication the basic integrated circuit (IC) is fabricated, layer upon layer, by processes such as photolithography, etching, and diffusion. It is the most complex process/stage of production, containing between 300 and 500 process steps which are performed by various types of complex production tools. Individual circuits are electrically tested during wafer probe, with the functioning ICs being placed in various types of packages during wafer assembly. Finally, the completed, packaged ICs are tested once more during final testing before being sent to the customer.

This paper is motivated by the wafer fabrication and final testing phases of semiconductor manufacturing. In addition to a number of other processing complexities, such as re-entrant flow, parallel machines, and machine setup requirements, the wafer fabrication process is a complex job shop characterized by both discrete and batch processing tools (Mason *et al.* 2002). Discrete processors process individual wafers or entire “lots” of 25 wafers one at a time, while batch processors multiple lots of wafers simultaneously.

Batch processing is common in a number of manufacturing sectors. Jobs or the “lots” are aggregated and processed simultaneously to increase production efficiency on machines called batch processing machines. In most of these scenarios the decisions regarding the formation of the batches, i.e., which job to put in which batch and then the subsequent scheduling of these formed batches are crucial decisions to be taken. In some cases batch processing times are the longest therefore the batch processing machine is often the “bottleneck” of the manufacturing unit. In order to bring down the cycle time of the product it is really important to schedule such batch processing machines because it is essentially the effective control of the bottleneck which leads to effective control of the overall system.

## **2. LITERATURE REVIEW**

Scheduling of batches is a common part of scheduling of wafer fabs and other parts of the semiconductor manufacturing like the testing stage. Most batch scheduling literature usually considers the problem as consisting of two sub-problems, one of batch formation i.e., the assignment of the jobs to the batches and the other the sequencing of the formed batches on a single batch processing machine such as the diffusion chamber.

Ahmadi *et al.* (1992) consider batching and scheduling in a two-machine flow-shop consisting of different combinations of batch and discrete processors. The objectives investigated are to minimize the makespan and sum of job completion times by developing heuristics that run in polynomial time. The heuristics vary according to the system configuration. For example, a simple sorting based heuristic solves a flow-shop problem when the configuration is a batch processor followed by a discrete processor where as a dynamic programming approach solves it when the configuration is discrete processor followed by the batch processor.

Uzsoy (1994) schedules a single batch processing machine with non-identical job sizes to minimize total completion time and makespan. Uzsoy (1994) proves that the problems are NP-hard and heuristics based on the bin-packing problem are developed for minimizing the makespan. A branch and bound approach is used to solve the total completion time problem. Lower bound for the problem is calculated by relaxing the problem to be one of jobs with unit size. Heuristics based on greedy algorithms are developed for this problem and it is found that these heuristics work well.

### *Multiple Orders per Job Batch Scheduling with Incompatible Jobs*

Uzsoy (1995) extended the above problem to deal with batching and scheduling of jobs belonging to incompatible families. The problem is motivated by the diffusion process in the wafer fabrication process. Static jobs, where all jobs are available at the same time, are considered and it is shown that there should be no partial batches formed and using this greedy algorithm based heuristics are developed to minimize the makespan, maximum lateness, and total weighted completion time and these methods are extended to the parallel machine environment. Efficient heuristics are developed for the minimizing the makespan and maximum lateness in the presence of dynamic job arrivals.

Mehta and Uzsoy (1998) consider minimizing the total tardiness on a batch processing machine in which the jobs belong to incompatible families. The processing time of the jobs is processing time of the family to which it belongs. Mehta and Uzsoy (1998) prove that the problem is NP-hard when the number of families and batch capacity is arbitrary and then go on to describe some characteristics of the optimal solution. A dynamic programming algorithm is developed in which the initial batch is formed by using a greedy algorithm. Batches are added from the set of unscheduled batches in a way so as to minimize the tardiness function value. A heuristic is developed by modifying the ATC dispatching rule so that it is applicable to batches and this rule is called BATC. A batch priority index calculated for each batch and then the batch with the highest priority index is scheduled. A simpler decomposition heuristic is also developed in which batches are scheduled according to the BATC rule and then the dynamic program is applied to a subset to get optimal batching and scheduling and this is done till all jobs are covered. The heuristics were found to give near optimal solutions in reasonable amount of computation time.

Dobson and Nambimadom (2001) investigate the problem of batching and scheduling in which the jobs belong to different families and jobs have different sizes therefore use different amount of the resource, which is a single batching machine. The processing time of the batch is independent of the size and the number of jobs in the batch. The objective is to minimize the mean weighted flow time. An integer programming formulation is given which is found to be NP-hard. Heuristics are developed to solve the problem based on the structure of the problem. An LP relaxation of the integer program gives lower bounds to the problem. It is found that it is easier to break up the problem into two problems, one of efficiently packing the jobs into a batch and the second of sequencing the batches. According to the authors, once a sequence is obtained then it is easier to efficiently pack the jobs into batches. Three heuristics are developed: a greedy heuristic, a knapsack heuristic and a generalized assignment heuristic. From computational study it is seen that the assignment heuristic and knapsack heuristic are better than the greedy heuristic and produce near optimal solutions.

The above research is the closest to the research work in this paper in the sense that both look at batch processing of jobs belonging incompatible families with non-identical job sizes. There some fundamental differences which should be emphasized:

1. The overall environment in which the scheduling is being done is different in the sense that this paper deals with the multiple orders per job scheduling problem.
2. In Dobson and Nambimadom (2001), there are two basic units: jobs and batches and the decision of which batch belong to which family is made by the optimization model. In this paper there are 3 basic units: orders, jobs and batches and orders belong to certain families. The decision regarding assigning the orders to the jobs and the formed jobs to the batches is made by the optimization model.
3. In the paper, there is no restriction on the number of jobs in the system but in our models and MOJ environment, the number of jobs/FOUPS is limited.

### 3. INTEGRATED BATCH FORMATION AND SCHEDULING IN SEMICONDUCTOR MANUFACTURING

The semiconductor manufacturing industry has recently adopted a new standard as regards the size of the basic wafer on which the circuits are fabricated. The silicon wafers are now 300mm in diameter and the new fabs are called 300mm wafer fabs. Earlier the basic size was 200mm in diameter. The increase in size has brought about an increase in weight of each wafer. Now, complete automation of the material handling systems, both inter-bay and intra-bay, is required. Due to this increase in size the industry has modified most of its equipment and also the material handling system. The lots of wafers are now transferred in units called front opening unified pods (FOUPS). FOUPS are containers that hold a certain number of 300mm wafers in an inert nitrogen atmosphere which protects the wafers from contamination. FOUPS are expensive entities and it is not economical to assign each order its own FOUF. It would also overload the automated material handling system. Therefore 300mm semiconductor manufacturers would need to group orders from different customers into one FOUF.

Qu (2004) has investigated the problem of fitting orders belonging to different customers into the same jobs so that the scheduling is done at the job level but the customer satisfaction is measured at the order level. The problem investigated in this paper is the next step in this research of multiple orders per job scheduling problem. The methods and principles of grouping the orders into jobs and then the jobs into batches so as to schedule them on a single batch processing machine with capacity constraints are investigated so as to minimize a due-date related performance measure (total weighted tardiness) which is a true measure of on-time delivery of orders to customers.

Figure 1 gives a good representation of the problem of interest. Let  $O = \{1..N\}$  be the set of orders with  $N$  being the number of orders. Each order  $o \in O$  has the following parameters associated with it:

1.  $s_o$ , the size of order  $o \in O$  in number of wafers. The size of the order determines the number of slots it occupies in its associated FOUF.
2.  $w_o$ , the weight or priority of order  $o \in O$ . The manufacturer may assign each order its own importance or priority and this parameter quantifies the importance.
3.  $d_o$ , the due-date of order  $o \in O$ . This parameter represents the day that the order must complete manufacturing in order to meet the delivery date of the order.
4.  $f_o$ , the family of order  $o \in O$ . Each order belongs to a particular family of product types. There are a fixed number of families and each order belongs to one of these families.

There is a capacity limit on the number of wafers that can fit into a FOUF and it is denoted by  $K$ . The customer order size  $s_o$  can be less than, equal to or greater than this limit. The size and configuration of FOUPS is governed by an internationally recognized standard. Typical values for  $K$  are 13 or 25. For an order  $o$  with  $s_o < K$ , there is the potential for it to be combined with other orders. If  $s_o > K$ , then a single order cannot be placed into the FOUF due to capacity constraints. This, in turn, results in order splitting. An important basis of the research is that the orders are not going to be split up between FOUPS, therefore we assume that for all orders  $s_o < K$ .

Now, the FOUPS are later combined into batches which then undergo batch processing operations such as diffusion or burn-in operations. There is an upper limit on the number of FOUPS that can be placed into a batch and it is denoted by  $C$ . The objective investigated is that of the total weight tardiness (TWT) of the orders. It is computed as  $TWT = \sum_{o \in O} w_o T_o$ . The parameter  $T_o$ , the tardiness of order  $o$ , is

calculated as  $T_o = \max(0, \delta_o - d_o)$ . The objective perfectly captures the issue of on-time delivery of the orders to the customers which is a crucial requirement of any business organization. This paper focuses on the case of incompatible job families. In this environment, the orders belonging to certain

## Multiple Orders per Job Batch Scheduling with Incompatible Jobs

families have to be grouped together into jobs in such a way that ensures that orders belonging to different families are not grouped into the same job. Essentially it is like assigning a job to a particular family. The formed jobs then have to be grouped together into batches and this grouping should again ensure that orders belonging to different families are not assigned to the same batch. Again it is like assigning a batch to a particular family. This batching is done for the diffusion operation. The processing time of the batch is a constant independent of the contents of the batch. The families of orders have constant processing times which are known and the processing time of the batches is the processing time of the family it is assigned to.

### 4. MIXED INTEGER PROGRAM

The following is a list of notation for the proposed optimization model:

$K$	Fixed capacity of the FOUP
$C$	Number of FOUPS/Jobs in a batch
$\rho_f$	Processing time of a single wafer of family type $f$ .
$\delta_o$	Completion time of order $o$ . The completion of the order is the completion time of the batch.
$\xi_b$	Completion time of the batch $b$ and it depends on the environment.
$I_f$	Index of orders $O$ which belong to family $f$ .
$F_m$	Set of Families
$B$	Set of Batches
$N_B$	Number of batches in the model = $F_m$
$F$	Number of jobs in the system
$O$	Set of Orders, $\{1..N\}$
$J$	Set of Jobs, $\{1..F\}$

Parameters of the orders have been defined in the previous section. An important point to mention is that with out loss of generality, it is assumed for both models, the batches are processed in increasing order of the index, i.e., the sequence of processing the batches on the batching machines is known *a priori*.

The decision variables used in this model are as follows:

$X_{ojb}$	1 if order $o$ is assigned to job $j$ is assigned to batch $b$ else 0
$Y_{jf}$	Integer, tracks which job $j$ belongs to which family $f$
$Z_{bf}$	Integer, tracks which batch $b$ belongs to which family $f$
$L_{jb}$	1 if job $j$ is in batch $b$ else 0

The model formulation is as follows:

$$\begin{aligned} \text{Min} \quad & \sum_{o \in O} w_o T_o \\ \text{s.t.} \quad & \sum_{o \in I_f} \sum_{b \in B} s_o X_{ojb} \leq K \quad \forall j \in J, f \in F & (1) \\ & \sum_{j \in J} \sum_{b \in B} X_{ojb} = 1 \quad \forall o \in I_f, f \in F & (2) \\ & Y_{jf} \geq X_{ojb} \quad \forall o \in I_f, j \in J, b \in B, f \in F & (3) \end{aligned}$$

## Multiple Orders per Job Batch Scheduling with Incompatible Jobs

$$\sum_j Y_{jf} \geq 1 \quad \forall f \in F \quad (4)$$

$$Z_{bf} \geq X_{ojb} \quad \forall o \in I_f, j \in J, b \in B, f \in F \quad (5)$$

$$\sum_b Z_{bf} \geq 1 \quad \forall f \in F \quad (6)$$

$$\sum_f Z_{bf} = 1 \quad \forall b \in B \quad (7)$$

$$L_{jb} \geq \sum_{o \in I_f} X_{ojb} / N \quad \forall j \in J, b \in B, f \in F \quad (8)$$

$$\sum_{b \in B} L_{jb} = 1 \quad \forall j \in J \quad (9)$$

$$\sum_{j \in J} L_{jb} \leq C \quad \forall b \in B \quad (10)$$

$$\xi_b \geq \rho_f Z_{bf} \quad \forall o \in I_f, j \in J, b \in B, f \in F \quad (11)$$

$$\xi_{b+1} \geq \xi_b + \rho_f Z_{b+1,f} \quad \forall j \in J, b \in [1 \dots N_{B-1}], f \in F \quad (12)$$

$$\delta_o \geq \xi_b - M \left( 1 - \sum_{j \in J} X_{ojb} \right) \quad \forall o \in I_f, b \in B, f \in F \quad (13)$$

$$T_o \geq \delta_o - d_o \quad \forall o \in I_f, f \in F \quad (14)$$

$$\delta_o \geq 0 \forall o \in O, \xi_b \geq 0 \forall b \in B$$

Constraints (1) and (2) are for the assignment of the orders to the jobs and to the batches. The first constraint ensures that the capacity of the job/FOUP is not exceeded by placing the orders into the jobs. The second constraint ensures that an order is assigned to a single job and a single batch and not split across the jobs and batches.

Constraints (3) and (4) are for ensuring that there is a family associated with each job, which depends on the orders present in the job. Due to capacity limitations, a single family may consist of more than a single job. Constraint (4) takes care of this issue.

Constraints (5), (6) and (7) are similar to the above constraints but they deal at the batch level to ensure that there is a unique family associated with each batch. Each batch can be associated with a single family and constraint (7) takes care of that. Constraints (8) and (9) describe the assigning of the jobs to the batches and together ensure that jobs are not split across batches. Constraint (10) is to check the fact that the capacity of the batch (described in terms of the number of jobs) is not exceeded.

The last 4 constraints (10-14) are for calculating the completion time of the batches, the completion time of the orders present in the batches and the tardiness of the orders. There is a constant processing time associated with each family and the processing time and completion time of the batch depends on the family it gets associated with and this depends on the association of the order with the families present in the batch.

## 5. EXPERIMENTAL DESIGN AND PRELIMINARY RESULTS

In order to investigate the performance of the proposed mixed integer programming (MIP) formulation of the multiple orders per job batch scheduling model, a number of experiments were

### Multiple Orders per Job Batch Scheduling with Incompatible Jobs

conducted using the experimental design described in Table 1. In the experiments, the penalty parameter big M was calculated as  $M = F * \max(\rho_f)$ . The numbers of FOUPS in the system, F, were calculated as

$$\max \left( \left\lceil \frac{F_m * N_F * \nu}{12\beta} \right\rceil + 1, F_m \right) \quad (15)$$

The weights in the instances were calculated using the discrete distribution *DU* [1, 15]. The above parameters were taken from Mason *et al.* (2004). The parameters related to due date calculations were taken from Mehta and Uzsoy (1998). Parameter *T* is the expected percentage of tardy jobs and *R* is the range parameter. A high value of *T* implies a lower mean due date and a high value of *R* means a higher variance among job due dates. For each experimental design factor combination, 10 problem instance replications were randomly generated and analyzed on a Pentium IV 3.00 GHz CPU with 1 GB RAM. The makespan used in Mehta and Uzsoy (1998) is calculated as:

$$C_{\max} = n_j \times m \times E[\rho_f] / B \quad (16)$$

where  $n_j$  = number of jobs/family,  $m$  = number of families,  $E[\rho_f]$  = expected processing time of families,  $B$  = batch capacity or number of jobs/batch. If we use a similar expression for our research problem, the expression that results is

$$C_{\max} = N_F \times F_m \times \text{avg}(\rho_f) / C \quad (17)$$

where the terms in the expression are from Table 1.

We note that for our MOJ batch scheduling research such a definition of makespan is inappropriate. This becomes clearer if we perform a dimensional analysis on the expression. In terms of base dimensions, the expression is

$$C_{\max} = \left( \frac{\#orders}{family} \right) \times (\# families) \times \left( \frac{time}{family} \right) / \frac{jobs}{batch} = \left( \frac{(\#order)(time)}{(job)} \right) \quad (18)$$

This is because the number of families equals the number of batches in our MOJ batch scheduling problem of interest. Therefore we see that the dimensions of makespan are not units of time. This difference occurs because; in this problem there are three elements (i.e., order, jobs, and batches) to be scheduled instead of two in the original problem (i.e., jobs and batches).

By looking at the expression we note that we need to multiply it by a  $(jobs/\#orders)$  term to get makespan in proper dimensional units. We note that this expression has an upper bound and lower bound based on the number of FOUPS in the system calculated in (15). The range for the makespan is:

$$C_{\max} = \left[ \left( \frac{F_m}{N_F F_m} \right) \left( \frac{N_F F_m \bar{\rho}_f}{C} \right), \left( \frac{F_m N_F \nu / 12\beta}{F_m N_F} \right) \left( \frac{N_F F_m \bar{\rho}_f}{C} \right) \right] \\ = \left[ \frac{F_m \bar{\rho}_f}{C}, \frac{N_F F_m \bar{\rho}_f \nu}{C(12\beta)} \right] \quad (19)$$

Once initial experiments were conducted to verify the value of the percentage of tardy orders that resulted when *T* was set to 0.3 and 0.6, respectively, the resulting order tardiness values were too high, as over 70% of the jobs were tardy in the “loose” due date case of *T*=0.3. As order due date setting is a function of estimated makespan, a new expression was derived for makespan. For small problem

instances, the sum of order family processing times is a good estimate of the makespan, as the minimum number of batches used is equal to the number of families in the instance. As the number of batches required per family increases, future research is ongoing to obtain better estimates for makespan in larger problem instances, as simply using  $makespan = \sum_{f \in F_m} \rho_f$  can adversely underestimate due-date range and result in an unnecessarily tight order due date range.

The computation time required to obtain the optimal solutions for six-, eight-, and nine-order MOJ batch scheduling instances is given in Table 2. We note that these times seem reasonable on average, but worst case solution time can be quite poor for such small problem instances. Therefore, heuristics must be developed that give near optimal solutions in reasonable amount of computation time.

## 6. HEURISTIC DEVELOPMENT AND ANALYSIS

Mason *et al.* (2004) develop heuristics based on the first fit decreasing approach to solve the single machine multiple orders per job scheduling problem. The problem is treated similar to the bin-packing problem and first fit decreasing based heuristics, which is known to work well for the bin-packing problem, are applied to the single item and single lot problems to get solutions which are 1% and 2% above optimal respectively. A similar approach is used in heuristic development for the multiple orders per jobs batch scheduling problem with incompatible families. The problem can be considered to be consisting of sub-problems of assigning orders to jobs and then assigning these formed jobs to batches in order to minimize the total weighted tardiness of the orders. Mason *et al.* (2004) have developed good heuristics for the first sub-problem. The sub-problem that needs to be addressed is to assign these formed jobs to the batches while keeping the constraint of incompatible families satisfied. The heuristic development consists of:

- Order Sequencing
- Order to Job Assignment
- Job to Batch Assignment

### 6.1 Order Sequencing

As a preliminary step, orders are sorted according to two methods:

- EDD: Sort the orders in non-decreasing order of order due-dates.
- WEDD: Sort the orders in non-increasing ratio of weight and due-date of each order.

### 6.2 Order-to-Job Assignment

The next step is to assign these sorted orders into the FOUPS/jobs available. The order to job assignment is performed using (Mason *et al.* 2004):

- FFD: This is an implementation of the first-fit decreasing method in which the sorted orders are placed, one by one, into the jobs such that the capacity constraint of the jobs is not violated. Starting from the top of the order list, the selected order is put into the first job with available capacity. Once job 1 is full, then we start at the top and go through the order list to feasibly assign the available orders in job 2 in order to fill it as much as possible and continue till all the orders have been placed.
- FFD\_AJS: While orders are assigned to jobs according to first-fit criteria, this method tries to balance the size of the formed jobs. The job size is calculated as the sum of sizes of all orders

assigned to the job. At each iteration, the average size of each job is calculated by dividing the sum of sizes of all orders yet to be placed in the jobs by the number of currently unfilled jobs. This gives the expected average job size for each remaining job. The orders are assigned in such a way that they do not violate the FOUN capacity limit and also not exceed the expected average job size. If the size of the current job being filled exceeds the expected average job size or no more orders remain, then we stop filling the current job and consider the next job.

### 6.3 Job-to-Batch Assignment

These formed jobs are now assigned to the batches which are processed in increasing order of their indices respectively. Since decision making involves assigning the formed jobs to the available batches, *aggregated* (from the orders) information needs to be computed for the formed jobs. The aggregated information used for this purpose is:

- $w_j$  = Weight of job, which is the sum of weights orders placed in the job ( $w_j = \sum_{o \in j} w_o \forall j \in J$ )
- $d_j$  = Due-date of the job, which is computed to be minimum of due-dates of orders present in the job ( $d_j = \min(d_o) \forall o \in j, j \in J$ )
- $jpt_j$  = Job processing time. It is equal to the processing time of the family the job belongs to.

Using the aggregated information, the jobs are assigned to the batches using:

- FFDJB1: Sort jobs in increasing order of  $d_j$  and place in first available batch. Use weight of the jobs to break ties.
- FFDJB2: Sort jobs in decreasing order of ratio of weight of job ( $w_j$ ) times the number of orders in the job to sum of due-dates of the orders present in the job ( $\sum_{o \in J} d_o$ ) and place the jobs in the first available batch.
- FFDJBATC: The ATC rule (Vepsalainen and Morton, 1987) is applied to the jobs to sort them and then place these sorted jobs into the first available batch. ATC parameters are calculated as follows:

$$C_{\max} = \sum \rho_f \quad (20)$$

$$\bar{C}_{\max} = \frac{C_{\max}}{F_m} \quad (21)$$

$$R = \frac{\max(d_o) - \min(d_o)}{C_{\max}} \quad (22)$$

If  $R \leq 0.5$ ,  $K_l = 4.5 + R$ , else  $K_l = 6 - 2 * R$  (Pinedo, 2002). The ATC index is calculated according to Mehta and Uzsoy (1998):

$$I_j(t) = \frac{w_j}{jpt_j} \left( \exp \left( \frac{- \sum_{o \in J} \max(d_o - jpt_j - t, 0)}{K_l * \bar{C}_{\max}} \right) \right) \quad (23)$$

## Multiple Orders per Job Batch Scheduling with Incompatible Jobs

The parameter  $t$  is updated after each iteration to the job processing time ( $jpt_j$ ) of the recent most job placed in the batch. In the numerator, all the orders belonging to job  $j$  are considered one by one and their individual “slack” is summed up. The job with the highest index value is assigned to the first available batch. The process is repeated till all jobs have been assigned to the batches.

For each of the above methods, the constraint of incompatible families is implemented at all levels. It is ensured that orders belonging to different families are not assigned to the same job and the jobs belonging to different families are not assigned to the same batch. We note that there are  $2*2*3=12$  combinations to be run on each instance and the combination which consistently finds near optimal solutions in reasonable amount of computation time would be considered as being an applicable heuristic for this problem. The performance ratio of solutions obtained by the heuristics to the optimal solutions obtained using the MIP are calculated ( $PR = \frac{TWT(H)}{TWT(OPT^*)}$ ), where  $TWT(H)$  is the total weighted tardiness value obtained by applying different combinations of simple heuristics  $H$  and  $TWT(OPT^*)$  is the solution obtained from the MIP. The different heuristic combinations are designated as shown in Table 3.

### 6.4 Experimental Results

From the selected  $PR$  results in Table 4, it can be seen that heuristic combinations  $H_3$ ,  $H_6$ ,  $H_9$ , and  $H_{12}$  give solutions that are 4% above optimal, on average. A given heuristic combination required less than one second to evaluate each of the problem instances under study. The other combinations in this table denote the worst case heuristic performance, while the missing heuristic combinations performed at a level between the best and worst cases. The average  $PR$  is the best when the number of orders per family ( $N_F = 4$ ) with the heuristic solution being 2% above optimal. At the time of this submission, the 12-order problem instance optimal solutions were not available. The  $PR$  is seen to increase to 4 % above optimal as we increase the number of families in the model from 2 to 3. The heuristics perform quite well for the factor level setting of  $T = 0.6$ , with the best heuristic producing a  $PR$  of nearly 1.0. However, the same heuristics give a 6% above optimal solution for parameter setting of  $T=0.3$ .

It is seen that combination of EDD, FFD1 and FFDBATC methods gave good results but the drawback of FFDBATC method lies in its sensitivity to the calculation of the look-ahead parameter. The look-ahead parameter was calculated using the order data in the instance. A grid search was performed to estimate appropriate values for the ATC rule’s look ahead parameter by running the combination  $H_3$  (the combination that gave the best heuristic solution) on the problem instances. Values of  $K_l = 2.5$  to  $K_l = 4$  all give solutions that are 3% above optimal, on average. When  $K_l = 3.5$ , the best performance results wherein average performance is 3.3 % above optimal and worst case performance is 5% above optimal.

## 7. CONCLUSIONS AND FUTURE WORK

A mixed-integer programming formulation for the multiple orders per job batch scheduling problem on a single batch processing machine is presented, as motivated by 300-mm semiconductor manufacturing diffusion operations wherein jobs that belong to the same family (product type) can be grouped together in a production batch. Experimental testing of this initial model formulation suggests

## *Multiple Orders per Job Batch Scheduling with Incompatible Jobs*

that while optimal solutions are achievable for this initial set of six to nine order problems within a reasonable amount of computation time, preliminary trends suggest the need for heuristic development to support the analysis of larger, more practical MOJ batch scheduling problems.

It can be inferred from the results that as we increase the number of families and the number of orders per family the computation time is increasing. Heuristics are developed that solve the problem in reasonable amount of computation time and give solutions that are near optimal. The heuristic approaches consist of sorting the orders according to a predetermined rule, placing these sorted orders in the available jobs and the placing these jobs in the available batches to minimize the total weighted tardiness of the orders. The combination of EDD (to sort the orders), FFD1 (to place the sorted orders into the jobs) and FFDBATC (to assign the formed jobs to batches) works efficiently to give solutions that are 3% (on-average) above optimal. The 12 orders results need to be incorporated in to the above obtained results to get a full set of results for the experiment.

In our future research efforts, we will be developing and testing a mixed integer programming formulation for the multiple orders per job batch scheduling problem with *compatible* batching, as well as extending this formulation to a multiple machine environment. These future research efforts will help us to continue along our path towards developing viable scheduling and dispatching approaches for MOJ environments as characterized by 300-mm semiconductor manufacturing.

## REFERENCES

- Ahmadi, J.H., Ahmadi, R.H., Dasu, S., Tang, C.S., 1992. Batching and scheduling jobs on batch and discrete processors, *Operations Research*, 40, 750-763.
- Chandru, V., Lee, C.Y., Uzsoy, R., 1993. Minimizing total completion time on batch processing machines. *International Journal of Production Research*, 31, 2097-2121.
- Devpura, A., Fowler, J.W., Carlyle, W.M., Perez, I., 2000. Minimizing total weighted tardiness on a single batch process machine with incompatible job families, *Symposium on Operations Research 2000, Dresden, Germany*, 366-371.
- Dobson, G., Nambimadom, R. S., 2001. The batch loading and scheduling problem, *Operations Research*, 49, 52-65.
- Kutanoglu, E., Tanrisever, F., Mason, S.J., 2004. Forming and scheduling jobs from multiple orders with different attributes: A computational study of special cases. *IERC conference 2004, Houston, TX, USA*.
- Knutson, K., Kempf, K., Fowler, J. W., Carlyle, M., 1999. Lot-to-order matching for a semiconductor assembly & test facility, *IIE Transactions*, 31, 1103-1111.
- Lee, C.Y., Uzsoy, R., Martin-Vega, L.A., 1992. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40, 764-775.
- Mason, S.J., Fowler, J. W. Carlyle, W. M., 2002. A modified shifting bottleneck heuristic for minimizing the total weighted tardiness, *Journal of Scheduling*, 5, 247-262.
- Mason, S.J., Qu, P., Kutanoglu, E., Fowler, J.W., 2004. The single machine multiple orders per job scheduling problem, *IIE Transactions*, being revised
- Mehta, S.V., Uzsoy, R., 1998. Minimizing total tardiness on batch processing machine with incompatible job families, *IIE Transactions*, 30, 165-178.
- Pinedo, M., 2002, *Scheduling Theory, Algorithms, and Systems*, Prentice Hall Inc., Upper Saddle River, New Jersey.
- Qu, P., 2004, *The Multiple orders per job scheduling problems*, PhD Dissertation, University of Arkansas, Fayetteville, AR.
- Uzsoy, R., Lee, C.Y., Martin-Vega, L.A. 1992. A review of production planning and scheduling models in the semiconductor industry. Part I: system characteristics, performance evaluation and production planning, *IIE Transactions*, 24, 47-59.
- Uzsoy, R., 1994. Scheduling a single batch processing machine with nonidentical job sizes, *International Journal of Production Research*, 32, 1615-1635.
- Uzsoy, R., 1995. Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*, 33, 2685-2708.

*Multiple Orders per Job Batch Scheduling with Incompatible Jobs*

Vepsalainen, A.P.J., Morton, T.E., 1987. Priority rules for job shops with weighted tardiness costs, *Management Science*, 33(8), 1035-1047.

*Multiple Orders per Job Batch Scheduling with Incompatible Jobs*

Table 1. Experimental Design

<b>Factors</b>	<b>Levels</b>	<b>Level Description</b>
Problem Type ( $P$ )	1	Incompatible Job Families
Number of Families ( $F_m$ )	2	2,3
Order Size ( $s_o$ )	2	Discrete uniform $DU\left[v - \frac{v+1}{2}, v + \frac{v+1}{2}\right]$ , where $v \in \{3,5\}$
FOUP Capacity ( $K$ )	2	$12\beta + 1$ , where $\beta \in \{1,2\}$
Capacity of Batches ( $C$ in terms of number of jobs)	2	2,3
Number of Orders ( $N_F$ ) per family	2	3,4
Due Date ( $d_o$ )	4	$DU\left[\mu - \frac{\mu R}{2}, \mu + \frac{\mu R}{2}\right]$ , where $\mu = \text{makespan} * (1 - T)$ Makespan = $\sum \rho_f$ $\rho_f =$ processing times of the families in the instance $T = 0.3, 0.6$ $R = 0.5, 2.5$
Processing Time per family ( $\rho_f$ )	1	2 with probability of 0.2 4 with probability of 0.2 10 with probability of 0.3 16 with probability of 0.2 20 with probability of 0.1

96 Unique Design Factor Combinations  
10 Problems per Design Factor Combination  
**960 Total Problems Instances Investigated**

*Multiple Orders per Job Batch Scheduling with Incompatible Jobs*

Table 2. Computation Time for MIP

		Time(secs)	Min	Max
Order Size ( $\nu$ )	(1,3,*,*,*,*,*)	4.5	0.0	55.2
	(1,5,*,*,*,*,*)	46.3	0.1	805.4
Size Factor ( $\beta$ )	(1,*,1,*,*,*,*)	49.9	0.1	805.4
	(1,*,2,*,*,*,*)	0.8	0.0	3.5
Number of Families ( $F_m$ )	(1,*,*,2,*,*,*)	4.0	0.0	187.8
	(1,*,*,3,*,*,*)	68.1	0.6	805.4
Orders per Family ( $N_F$ )	(1,*,*,*,3,*,*,*)	34.4	0.0	805.4
	(1,*,*,*,4,*,*,*)	7.4	0.0	137.8
Batch Capacity ( $C$ )	(1,*,*,*,*,2,*,*)	25.2	0.0	805.4
	(1,*,*,*,*,3,*,*)	25.5	0.0	739.7
% Tardy ( $T$ )	(1,*,*,*,*,*,0.3,*)	16.9	0.0	524.6
	(1,*,*,*,*,*,0.5,*)	33.9	0.0	805.4
Due Date Range ( $R$ )	(1,*,*,*,*,*,*,0.5)	29.9	0.0	805.4
	(1,*,*,*,*,*,*,2.5)	20.8	0.0	677.8
Overall	(1,*,*,*,*,*,*,*)	25.4	0.0	805.4

*Multiple Orders per Job Batch Scheduling with Incompatible Jobs*

*Table 3. Heuristic Combinations for Evaluation*

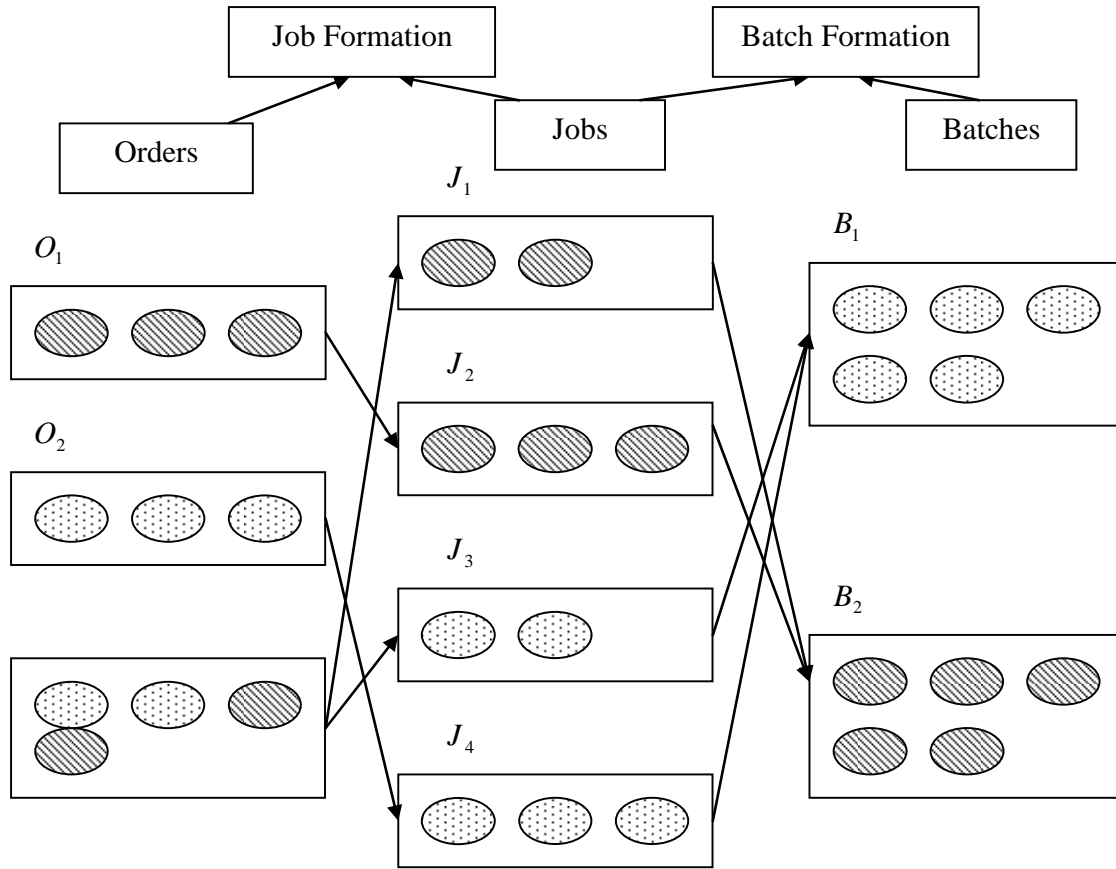
Designation	Sort Order	Order to Job	Job to Batch
$H_1$	EDD	FFD	FFDJB1
$H_2$	EDD	FFD	FFDJB2
$H_3$	EDD	FFD	FFDJBATC
$H_4$	EDD	FFD_AJS	FFDJB1
$H_5$	EDD	FFD_AJS	FFDJB2
$H_6$	EDD	FFD_AJS	FFDJBATC
$H_7$	WEDD	FFD	FFDJB1
$H_8$	WEDD	FFD	FFDJB2
$H_9$	WEDD	FFD	FFDJBATC
$H_{10}$	WEDD	FFD_AJS	FFDJB1
$H_{11}$	WEDD	FFD_AJS	FFDJB2
$H_{12}$	WEDD	FFD_AJS	FFDJBATC

Multiple Orders per Job Batch Scheduling with Incompatible Jobs

Table 4. Best and Worst Case Heuristic PR Results

	PR						
	$H_3$	$H_4$	$H_6$	$H_7$	$H_9$	$H_{10}$	$H_{12}$
Order Size ( $\nu$ )							
(1,3,*,*,*,*,*)	1.029	1.347	1.033	1.336	1.029	1.341	1.033
(1,5,*,*,*,*,*)	1.044	1.409	1.048	1.377	1.044	1.406	1.047
Size Factor ( $\beta$ )							
(1,*,1,*,*,*,*)	1.039	1.376	1.045	1.35	1.039	1.369	1.044
(1,*,2,*,*,*,*)	1.034	1.379	1.036	1.363	1.034	1.378	1.036
Number of Families ( $F_m$ )							
(1,*,*,2,*,*,*)	1.031	1.335	1.035	1.302	1.031	1.331	1.034
(1,*,*,3,*,*,*)	1.048	1.463	1.051	1.464	1.048	1.459	1.051
Orders Per Family ( $N_F$ )							
(1,*,*,*,3,*,*,*)	1.042	1.386	1.046	1.380	1.042	1.381	1.045
(1,*,*,*,4,*,*,*)	1.026	1.361	1.030	1.310	1.026	1.360	1.029
Batch Capacity ( $C$ )							
(1,*,*,*,*,2,*,*)	1.046	1.368	1.050	1.364	1.046	1.365	1.049
(1,*,*,*,*,*,3,*,*)	1.027	1.388	1.031	1.348	1.027	1.382	1.031
% Tardy ( $T$ )							
(1,*,*,*,*,*,0.3,*)	1.064	1.410	1.067	1.379	1.064	1.408	1.067
(1,*,*,*,*,*,0.6,*)	1.009	1.346	1.014	1.333	1.010	1.340	1.013
Range ( $R$ )							
(1,*,*,*,*,*,*,0.5)	1.030	1.380	1.036	1.364	1.030	1.373	1.035
(1,*,*,*,*,*,*,2.5)	1.044	1.376	1.045	1.349	1.044	1.375	1.045
Overall							
(1,*,*,*,*,*,*,*)	1.037	1.378	1.040	1.356	1.037	1.374	1.040

*Multiple Orders per Job Batch Scheduling with Incompatible Jobs*



*Figure 1.* The multiple orders per job batch scheduling problem