Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

SCHEDULING BATCH PROCESSING MACHINES IN COMPLEX JOB SHOPS

Kasin Oey
Scott J. Mason

Department of Industrial Engineering
4207 Bell Engineering Center
University of Arkansas
Fayetteville, AR 72701, U.S.A.

ABSTRACT

This paper considers a complex job shop problem with re-entrant flow and batch processing machines. A modified shifting bottleneck heuristic (MSB) is considered for generating machine schedules to minimize the total weighted tardiness. We observe that the MSB could produce infeasible schedules where cyclic schedules are found. A cycle elimination procedure is proposed to remove the possibility of the MSB generating cyclic schedules in the solution.

1 INTRODUCTION

Semiconductor manufacturing is one of the fastest growing industries in the world today--the market demand for semiconductors is increasing. The industry has grown since mid July 1999 and reached the highest total sales of $204 billion at the end of 2000 (Semiseek News 2001). Even though the recent US economy is slowing down the industry, most analysts still believe that the industry will rebound by late 2001 or sometime in 2002 (Silicon Strategies 2001). The demand for semiconductors is no longer based on PC sales but also communication and information devices that have become necessary tools of today (Smith 2001). The high competition between semiconductor manufacturers makes customer satisfaction very important. If a semiconductor manufacturer cannot deliver an order on time, customers will try to find other manufacturers that are more reliable. Therefore, the timely completion of orders is a high priority.

There are several performance measures that can be used to evaluate due-date based completion time of orders in a manufacturing facility, such as lateness and tardiness. The lateness of job j, L_j, is defined as the difference between the completion time of a job (C_j) and the job's due date (d_j). Tardiness (T_j) is the maximum of L_j and zero. A job may also have a certain weight or priority to represent the importance of the job. One of the performance measures that relates to jobs that have importance levels is weighted tardiness (w_jT_j). Since "real-world" jobs usually have levels of importance, weighted tardiness is a useful performance measure for evaluating a semiconductor manufacturer's delivery performance.

2 COMPLEX JOB SHOPS

A classical job shop typically contains n jobs where each job follows a predetermined route. The object of the classical job shop scheduling problem is to schedule all jobs in such a way to optimize the value of a given performance measure of interest. Each route can consist of several processing steps. A process step is always associated with a certain machine. Each job in a job shop can have a process step that requires the same machine as other jobs. There could be a maximum of n jobs competing for each machine in a job shop problem without re-entrant flow (wherein jobs visit the same machine multiple times). A job shop characterized by n jobs being processed on m machines will have $(n!)^m$ possible schedules. Therefore, the scheduling of a job shop is not an easy task.

A complex job shop has far more complicating issues compared with a classic job shop. An example of a complex job shop is a wafer fabrication facility ("wafer fab"). The number of processing steps in a wafer fab can vary between 300 up to 500 steps for each job route (product type). It is typical to use the term 'tool group' (TG) instead of machine in wafer fab environment, as a TG is a group of machines that has the same operating characteristics. A complex job shop usually has parallel machines operating within a TG, often with more than one machine available for processing. The possibility of sequence-dependent setup times also adds to the complexity of the wafer fab scheduling problem. The sequence of the processes that share a TG determines the total setup times required in the problem. Different sequences of processes could result in different setup times. For example, changing the dopant ion from boron to phosphorus in an ion implanter may require more time than changing from boron to arsenic. A

"good" schedule seeks to minimize total setup time, which also minimizes the total completion time of jobs.

Re-entrant flow processes also appear in the wafer fab environment where a job goes through the same TG more than once in its route. Re-entrant flow expands the number of possible job schedules on the same TG, thereby resulting in a greater solution space. The possibility of batching different jobs together for processing on a certain TG increases the complexity of the wafer fab scheduling problem. Jobs are batched together when they have the same job ID or the same recipe name. There is also a maximum number of jobs (b) that can be batched together at one time due to the tool's physical capacity. However, it is possible to batch less than b jobs, which makes the number of batch combinations to be large. For a job shop that has n jobs with b jobs as the maximum batch size, there are $O(n^b)$ batch possibilities.

Another parameter that can be used in batching jobs is batch horizon (Mason *et al.* 2001). Batch horizon defines the amount of time a tool will be held idle in order to form a "fuller" batch. If a job is ready within the batch horizon then that job can be batched together with other jobs that meet the same requirement. Considering all of the processing complexities described above, the wafer fab scheduling problem has too many solution possibilities that complete enumeration is not possible to be performed under a reasonable time. This makes sense, as the classical job shop scheduling problem is unsolvable. Therefore, researchers often employ heuristic approaches to attack these problems.

3 PREVIOUS WORK

Kubiak *et al.* (1996) discussed the use of shortest processing time job order and a dynamic programming algorithm to minimize the total flow time in a job shop. The job shop problem that they investigated has n jobs and k machines where each job enters the first machine (the hub) k times. Each job has the same route that visits alternately the first machine, M_1 (the hub) and M_2, M_3, …, M_k. Furthermore, there could be a setup time between two operations in M_1. While their approach seems to work well for this problem, there was not any discussion about how batch possibilities are handled.

Hwang and Sun (1997) explored a similar problem with sequence-dependent setup time. Their problem involves a two-machine flow shop in which jobs are processed more than once on the first machine. Using a modified dynamic programming approach, they were able to minimize the makespan of jobs. Nose *et al.* (1999) tried to solve the job shop scheduling problem using a genetic algorithm. The job shop problem that they covered involves re-entrant flows of job. Nevertheless, none of the above works include a discussion on batching processing machines.

Chandru *et al.* (1993) discussed single and parallel batch machines as a separate process from the job shop problem. Uzsoy (1995) described an approach to schedule batch machines with non-identical job families. In the batch machines problem with non-identical job families, each job has a family name where only jobs with the same family can be batched together. This is similar to the recipe concept in semiconductor fabrication. Lee and Uzsoy (1999) covered the single batch machine problem with dynamic job arrivals. None of these papers integrate the batch process within a job shop environment. Besides, only one batch process was being considered while there is more than one batch-processing step in wafer fab job shop problem.

Ahmadi *et al.* (1992) discussed a two-step process that has at least one batching machine. Still, the two-step process is a simple model compared to the wafer fab scheduling problem. While there has been a lot of discussion about batching and re-entrant flow processes separately, only a few integrate both processes in a single scheduling problem. Roobeek (1997) used a fuzzy logic based approach called A Better Choice in solving the wafer fab job shop problem. Mason *et al*. (2001) covered a complex job shop problem with re-entrant flow and batch possibilities. The wafer fab was taken as an example of the complex job shop problem. Mason *et al.* applied a modified shifting bottleneck heuristic (MSB) as an approach to the wafer fab job shop problem.

4 MODIFIED SHIFTING BOTTLENECK

The original shifting bottleneck heuristic developed by Adams *et al.* (1988), decomposes a job shop problem into several single machine subproblems. The subproblems are solved iteratively, with each subproblem involving one machine and a number of process steps (jobs) to be performed on that particular machine. Each job has a release (ready) time and due date that are determined by the sequence of the processes on other machines. The heuristic applies a certain "subproblem solution procedure" (SSP) to get a good solution for the subproblem. For example, the earliest due date (EDD) dispatching rule could be used as a SSP to schedule jobs based on the due date of each job.

The output of a SSP is a schedule that contains all the jobs performed on a particular machine. After each subproblem is solved, machine criticality measures (MCM) are calculated to determine which machine is a critical/bottleneck machine that has to be scheduled first. Then, the heuristic removes the chosen (scheduled) machine from evaluation in subsequent iterations and repeats the whole process. Table 1 gives a summary of how the SB heuristic works.

Table 1: Description of the SB Heuristic for Minimizing TWT in Complex Job Shops (Mason *et al*. 2001)

Step 1.	Let M = the set of all m machines and M_0 = the set of machines that have been sequenced or scheduled. Set $M_0 = \varnothing$.
Step 2.	Identify and solve the subproblems for each machine $i \in M \setminus M_0$.
Step 3.	Identify a critical or bottleneck machine $k \in M \setminus M_0$.
Step 4.	Sequence machine k using the subproblem solution from Step 2. Set $M_0 = M_0 \cup \{k\}$.
Step 5.	Optional: Reoptimize the schedule for each machine $m \in M_0$, considering the newly added disjunctive arcs for machine k.
Step 6.	If $M = M_0$, stop. Otherwise, go to Step 2.

The shifting bottleneck heuristic uses a disjunctive graph to illustrate the job shop problem. A disjunctive graph describes the processing step of each job as a node where each node within the same job is connected with a solid arc that represents a precedence constraint between the job processes (conjunctive arc). Disjunctive arcs (dashed arcs) are used to connect nodes with different jobs that are processed on the same machine. Each pair of nodes that share the same machine has two disjunctive arcs that represent the possible precedence constraints.

For example, a pair of nodes (2,1) and (2,2) that share a same machine has two possible precedence constraints, either processing node (2,1) before node (2,2) or *vice versa*. When a schedule for that particular machine is determined, each pair of nodes that share the machine will only have one disjunctive arc (made conjunctive after scheduling the machine) that shows the precedence constraint between each pair nodes. The lengths of the conjunctive arcs and the disjunctive arcs are the processing times for the nodes that the arcs leave. Figure 1 shows an example of a classical job shop disjunctive graph.

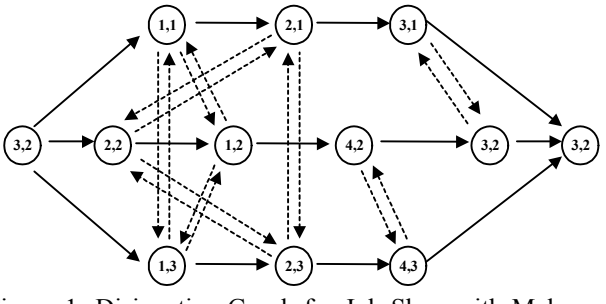

Figure 1: Disjunctive Graph for Job Shop with Makespan Objective (Pinedo and Chao 1999)

Mason *et al*. (2001) adapted the disjunctive graph by adding the re-entrant processes and 'dummy' nodes that represent all possible batch combination of jobs that meet the batching requirements (job type, recipe, etc.). The modified disjunctive graph represents a complex job shop where the term "tool group" is commonly used to describe a number of identical machines. Figure 2 displays the modified disjunctive graph with 3 jobs and 4 tool groups. For the sake of clarity, only one batch combination is displayed.

The arcs leaving the original nodes are not associated with the batch processing time. Instead, the arcs leaving the dummy node are associated with the batch processing time as indicated by P_{batch} in the figure. The arcs that leave the original nodes will have zero processing time, which represents the time required for batch formation.

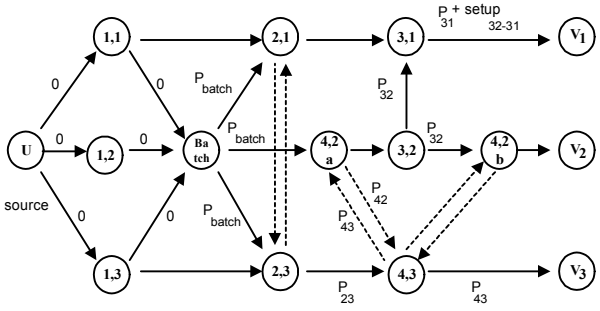

Figure 2: Modified Disjunctive Graph for Job Shop with TWT Objective (Mason *et al*. 2001)

In the classic job shop disjunctive graph, pairs of disjunctive arcs will connect processes (operations) that share the same tool group. The disjunctive arcs will form a clique in the graph (i.e., a complete sub-graph). In a job shop with re-entrant flow, a particular job can enter the same tool group more than once. Therefore, a clique will not be formed when re-entrant flow is present. Figure 2 shows a job shop problem where job 2 has a re-entrant process. Nodes (4,2a), (4,2b), and (4,3) are connected by disjunctive arcs. However, nodes (4,2a) and (4,2b) are not connected by disjunctive arcs. Node (4,2b) requires node (4,2a) to be processed first due to job 2 process flow requirements (process routing). Therefore, a conjunctive arc is used instead.

The length of the arcs associated with a tool group that has a sequence-dependent setup is not equal to the processing time of the associated node as it is in the classic disjunctive graph, as an additional setup time is required. For example, if node (3,1) in Figure 2 is scheduled after node (3,2) then the length of the arc leaving node (3,1) is equal to processing time of node (3,1) + setup time required to change the tool group's configuration from the processing requirements at node (3,2) to the requirements at (3,1). The length of the arc leaving node (3,2) will be equal to the processing time of node (3,2) (see Figure 2). Lastly, three dummy nodes are applied in the graph instead of one as in the classic job shop disjunctive graph. As the total weighted tardiness (TWT) objective requires information on the completion time for each job, a separate node for each job is required to represent the ending node for each job (Pinedo and Chao 1999).

A tool group is often made up of a number of identical machines (parallel machines). Figure 3 displays how the parallel machines condition is applied in the modified disjunctive graph. Initially, the disjunctive graph does not show the existence of parallel machines. After the parallel machine tool group is scheduled, conjunctive arcs that represent the precedence constraints between the nodes replace the disjunctive arcs pairs. In the parallel machines disjunctive graph, not all of the nodes that share the same tool group have a conjunctive arc that connects them. Only those that are scheduled in the same machine within the tool group will have conjunctive scheduling arcs.

For example, Figure 3 shows nodes (2,1), (2,2a), (2,2b), and (2,3) that share the same TG that has two identical machines. Initially all nodes are connected by pairs of disjunctive arcs. After a schedule for the particular TG is found, nodes (2,1) and (2,2b) are processed on the same machine, while nodes (2,2a) and (2,3) are processed on the other machine. Therefore, nodes (2,1) and (2,2b) are connected with a conjunctive arc and so are nodes (2,2a) and (2,3). However, no arc connects nodes (2,2a) and (2,1) or nodes (2,1) and (2,3) or nodes (2,3) and (2,2b).

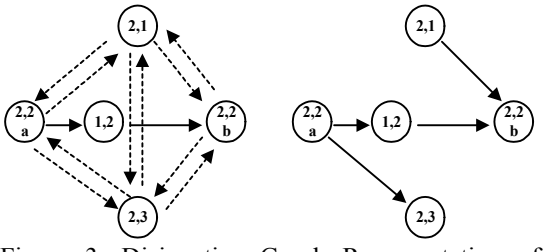

Figure 3: Disjunctive Graph Representation of Parallel Machine Scheduling (Mason *et al*. 2001)

The solution for the complex job shop problem will have a schedule for each tool group. A schedule is feasible when no cycles are present in the disjunctive graph (i.e., the schedule contains each node exactly once). A cycle is a sequence that starts and ends at the same point in the graph. A feasible schedule for a job shop problem in Figure 2 can be seen in Figure 4. Note that the arc lengths associated with each conjunctive scheduling arc have been updated.

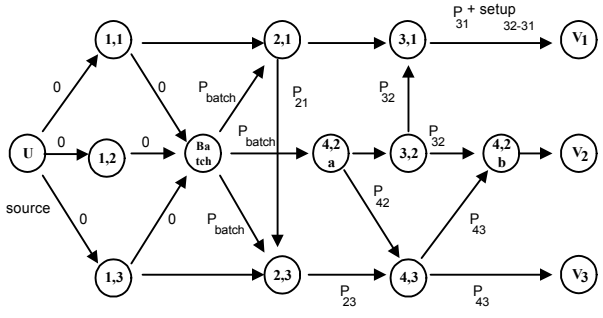

Figure 4: A Feasible Solution for Complex Job Shop Problem with TWT Objective

5 PROBLEM DESCRIPTION

The MSB approach developed by Mason *et al*. (2001), was found to have the possibility of getting an infeasible (cyclic) solution due to the existence of delayed precedence constraints and the dummy batch nodes that are used in the heuristic. The delayed precedence constraints are constraints that are caused by the scheduling of other tool groups (Balas *et al.* 1995). The delayed precedence constraints are represented by additional conjunctive arcs in the graph due to the other tool groups' schedules.

Figure 5 gives an example of a delayed precedence constraint. Before a schedule is found, node 3 does not have any precedence constraint. When a schedule is applied, conjunctive arcs are added to the graph which makes a precedence constraint exist between nodes 1 and 3 with a length of $P_1 + P_2$ unit time. The precedence constraint enforces node 1 must be performed at least $P_1 + P_2$ time units earlier than node 3.

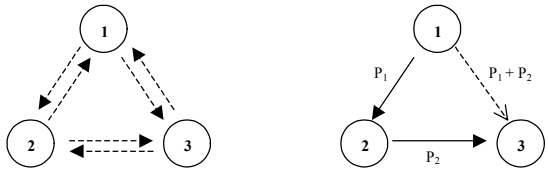

Figure 5: An Example of a Delayed Precedence Constraint

Assume $i < j$. The additional arcs that go from step j to an earlier step i (going backward in a disjunctive graph) produce cyclic paths when all batching possibilities are considered. The dummy batch nodes in the disjunctive graph connect all jobs that meet batching criteria requirements, thereby creating paths between jobs (see Figure 2). These paths, together with the delayed precedence constraints, can form a cyclic path that results in an infeasible solution.

Figure 6 shows an example of an initial disjunctive graph of a complex job shop with three jobs, three process flows, and two TGs (AB and C) with one machine on each TG. The disjunctive arcs are not shown for the sake of clarity. Only dummy batch nodes (nodes 20 and 21) that represent the possibility of batching job 1 and 3 together are presented in the graph. At this point, there is not any TG scheduled in the complex job shop.

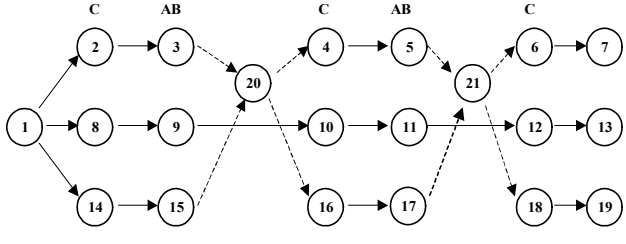

Figure 6: Initial Disjunctive Graph with Batch Possibility of Job 1 and Job 3

To show how the MSB approach can result in infeasible schedules, let the solution for TG C's schedule be 2-8-4-10-6-12-14-16-18. Figure 7 shows TG C's schedule. As TG C's schedule is applied, arcs are added to the disjunctive graph to represent the schedule. The additional arcs create delayed precedence constraints between some nodes. For example, there is not any precedence constraint between nodes 3 and 15 before the TG C schedule is applied. After the schedule is applied to the graph, a precedence constraint appears between nodes 3 and 15. Before node 15 is processed, the following sequence of nodes has to be performed first: 3, 4, 5, 6, 10, 11, 12, and 14.

The additional arc between nodes 12 and 14 is going backward (going to an earlier step) in the graph. The combination of this arc and the possibility of batching jobs 1 and 3 together creates a cyclic path. For example, the possibility of batching jobs 1 and 3 (represented by nodes 20 and 21) would produce the following cyclic schedules: 6-12-14-15-20-4-10-6 and 6-12-14-15-20-16-17-21-6. Note that in each cyclic schedule case, the starting and ending nodes of the schedule are the same (node 6).

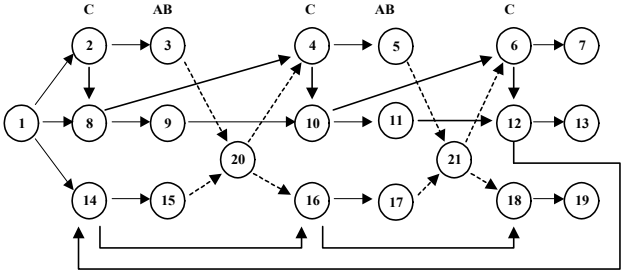

Figure 7: Disjunctive Graph After the Implementation of TG C

6 PROPOSED SOLUTION

Mason *et al.*'s (2001) MSB heuristic only results in an infeasible solution in cases where a non-batch TG is scheduled before batch-TGs are scheduled. The non-batch TG schedule forms delayed precedence constraints that, together with the dummy batch nodes, create the cyclic paths. If the dummy batch nodes 20 and 21 are removed in Figure 7, in other words, eliminating the possibilities of batching job 1 and job 3, no cyclic paths will be formed. A key to the solution is to find which batch nodes are feasible and which ones are not due to the possibility of forming cyclic paths.

The cyclic paths are related to a node *a* that is connected by a conjunctive arc to node *b* that has a lower step number, and the previously mentioned dummy batch nodes. Nodes that precede *a* and nodes that succeed *b* also contribute in creating cyclic paths. For example, node 12 and 14 in Figure 7 are *a* and *b* respectively. Job 2 and 3 are the job numbers associated with *a* and *b*. Therefore, no batching possibilities between job 2 and job 3 are allowed otherwise cyclic paths will be produced (12-14-15-dummy batch node-10-11-12 and 12-14-15-16-17-dummy batch node-12). Node 6 precedes node 12 and its associated job number is 1. Batching possibilities between job 1 and job 3 should not be considered otherwise cyclic paths will be formed (6-12-14-15-20-4-10-6 and 6-12-14-15-20-16-17-21-6).

However, the solution is not as easy as removing the batch possibilities of certain jobs on all batch steps. There might be a possibility of not batching certain jobs on a particular batch step and yet, the same jobs can be batched on other batch step. The positions of nodes relative to *a* and *b* in the disjunctive graph play an important role in finding the feasibility of batching jobs on a particular batch step. However, it is difficult to get the relative positions of nodes in a job shop problem.

A flow shop is a simplified version of a job shop as all jobs follow the same route. Similarly, a flexible flow shop provides for the existence of parallel machines, as opposed to the single-machine flow shop environment. Jobs in a flow shop have a fixed, common number of processing steps. Therefore, it is possible to establish each node's relative position according to the step number and the job number in the disjunctive graph. The disjunctive graph (not including the dummy nodes) can be represented as an X-Y matrix where X is the step number and Y is the job number.

Using the matrix, it is possible to check for nodes that will form a cycle if they are batched. Therefore, the job shop problem is modified into a flexible flow shop problem using dummy nodes to fill in the space in the matrix where a particular step is not included in a particular job route. Table 2 and 3 show an example how a complex job shop problem can be modified into a flexible flow shop problem. Each step has an associated TG type and a recipe name. Further, only jobs that have the same recipe name can be batched together.

Table 2: Job Shop Problem

Job	Step1	Step2	Step3
1	AB(R1)	E	-
2	CD	AB(R1)	-
3	CD	AB(R2)	AB(R1)

Table 3: Flow Shop Problem

Job	Step1	Step2	Step3	Step4
1	dummy	dummy	AB(R1)	E
2	CD	dummy	AB(R1)	dummy
3	CD	AB(R2)	AB(R1)	dummy

Based on the observations above, a general rule is proposed to eliminate the cyclic possibilities in the complex job shop scheduling problem. Define the following variables:

q = total number of jobs
n = total number of steps

m_i =TG at step i (i = 1, 2, …, n)
σ_{ij} =node with step i, job j (i = 1, 2, …, n ; j = 1, 2, …, q)
π_{ij} =predecessor node with step i, job j
ν_{ij} =successor node with step i, job j

The cycle elimination procedure can be described as follows:

Step 1. Modify the complex job shop problem into a flexible flow shop problem
Step 2. For each arc that represents $\sigma_{ij} \rightarrow \sigma_{kl}$ ($k < i, j \neq l$) on the current TG schedule, find all π_{cd} of σ_{ij} ($c > i$) and ν_{op} of σ_{kl} ($o < k$).
Step 3. Find σ_{rs} ($s = j$) and σ_{rv} ($v = l$), for each r ($i < r < k$, m_r = batch TG).
Step 4. For each π_{cd} ($c > r$, $d \neq j$) of σ_{ij}, find σ_{ra} ($a = d$), (m_r = batch TG).
Step 5. For each ν_{op} ($o < r$, $p \neq l$) of σ_{kl}, find σ_{rb} ($b = p$), (m_r = batch TG).
Step 6. For each r ($i < r < k$, m_r = batch TG), can't batch:
1. σ_{rs} with σ_{rv}
2. σ_{ra} with σ_{rv} (if σ_{ra} is found)
3. σ_{rs} with σ_{rb} (if σ_{rb} is found)
4. σ_{ra} with σ_{rb} (if σ_{ra} and σ_{rb} are found)

Consider Figure 8 where a job shop problem with four jobs has been modified into a flexible flow shop problem. The triangle nodes represent the dummy nodes that are added to form the flow shop. TG C, with a single machine, has been scheduled as the graph describes. The batch possibilities are represented by a simple box for the sake of clarification. In this problem, all jobs can be batched together, as they share the same recipe.

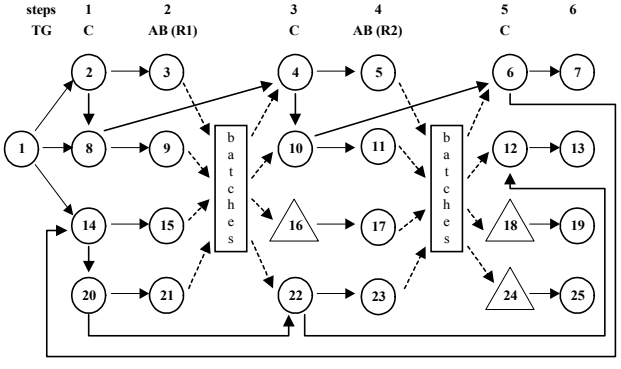

Figure 8: Example of Modified Complex Job Shop

The cycle elimination procedure is employed as follows:

Step 1. The complex job shop has been converted to a flexible flow shop in Figure 8.
Step 2. There is only one arc that represents $\sigma_{ij} \rightarrow \sigma_{kl}$ ($k < i$, $j \neq$ l) that is the arc that goes from node 6 (σ_{51}) to node 14 (σ_{13}).
Step 3. m_2 and m_4 are batch TG. $\sigma_{rs} = \sigma_{21}$ (node 3) and σ_{41} (node 5), $\sigma_{rv} = \sigma_{23}$ (node 15) and σ_{43} (node 17).
Step 4. π_{cd} ($c > r$, $d \neq j$) of $\sigma_{51} : \pi_{32}$ (node 10)
Only $r = 2$ applies in this case since $r = 4 > c = 3$
$\pi_{32} \rightarrow \sigma_{ra} = \sigma_{22}$ (node 9)
Step 5. ν_{op} ($o < r$, $p \neq l$) of $\sigma_{13} : \nu_{14}$ (node 20) and ν_{34} (node 22). ν_{14} and ν_{34} have the same job number and $o = 1$ in $\nu_{14} < r = 2$ and 4 , only ν_{14} needs to be considered. $\nu_{14} \rightarrow \sigma_{rb} = \sigma_{24}$ (node 21) and σ_{44} (node 23)
Step 6. On step $r = 2$, m_2 = batch TG, can't batch:
1. σ_{21} with σ_{23}
2. σ_{22} with σ_{23}
3. σ_{21} with σ_{24}
4. σ_{22} with σ_{24}

On step $r = 4$, m_4 = batch TG, can't batch:
1. σ_{41} with σ_{43}
2. σ_{41} with σ_{44}

Table 4 shows the cyclic paths that are removed by the cycle elimination procedure.

Table 4: Cyclic Paths List

batch removed	associated nodes	cyclic paths
σ_{21} , σ_{23}	3 , 15	6-14-15-BN-10-6
σ_{22} , σ_{23}	9 , 15	6-14-15-BN-10-6
σ_{21} , σ_{24}	3 , 21	6-14-20-21-BN-4-5-6
σ_{22} , σ_{24}	9 , 21	6-14-20-21-BN-10-6
σ_{41} , σ_{43}	5 , 17	6-14-15-16-17-BN-6
σ_{41} , σ_{44}	5 , 23	6-14-20-22-23-BN-6

The procedure also works for the parallel machines case where a TG consists of more than one machine. Parallel machines only change the relationship (path existence) between nodes in the disjunctive graph. Therefore, it will not affect the behavior of the procedure since the procedure will take any relationship condition. In fact, the parallel machines case has less probability of having cyclic paths as compared to the single machine case. Because nodes that are connected in the single machine problem might not be connected in the parallel machines problem (see Figure 3), the number of paths between nodes is reduced, thereby reducing the probability of having cyclic paths in the disjunctive graph.

7 CONCLUSION AND FUTURE RESEARCH

Semiconductor manufacturing is one of the fastest growing industries in the world today. The high competition between semiconductor manufacturers makes customer satisfaction very important. Therefore, the timely completion of orders is a high priority. A recently developed modified Shifting Bottleneck (MSB) scheduling approach to the complex job shop problem was shown to produce cyclic schedules due to the combination of delayed precedence constraints and dummy batching nodes. A cycle elimination procedure was developed for the MSB approach to promote cycle-free schedules.

The arcs that represent the delayed precedence constraints, particularly ones that go to earlier steps in the disjunctive graph, and dummy batch nodes in the MSB heuristic can cause the heuristic to obtain an infeasible solutions. The cycle elimination procedure is based on the characteristics of a flexible flow shop problem, which is a simplified version of the complex job shop problem. Batch node feasibility is evaluated in the procedure using the information on non-batching TG schedule that creates the delayed precedence constraints.

Another idea to avoid infeasible solutions is to schedule batch TGs first before scheduling other non-batch TGs. In this way, the batch nodes are fixed and the schedule of other TGs are based on the fixed batch nodes therefore eliminating the possibility of having cyclic paths. This approach does not count the MCM for batch TG that could lead to a worse solution quality. Mason (2000) has applied this on a small model and the result suggests that it is beneficial. However, the approach needs to be tested on larger problems to confirm its effect on the overall solution quality.

ACKNOWLEDGMENTS

Scott Mason is partially supported by the Semiconductor Research Corporation (2001-NJ-880).

REFERENCES

Adams, J., E. Balas, and D. Zawack. 1988. The shifting bottleneck procedure for job shop scheduling. *Management Science* 34: 391 – 401.

Ahmadi, J. H., R. H. Ahmadi, S. Dasu, and C. S. Tang. 1992. Batching and scheduling job shop on batch and discrete processors. *Operations Research* 40: 750 – 763.

Balas, E., J. K. Lenstra, and A. Vazacopoulos. 1995. The one-machine problem with delayed precedence constraints and its use in job shop scheduling. *Management Science* 41: 94 – 109.

Chandru, V., C. Y. Lee, and R. Uzsoy. 1993. Minimizing total completion time on batch processing machines. *International Journal of Production Research* 31: 2097 – 2121.

Hwang, H. and J. U. Sun. 1997. Production sequencing problem with reentrant work flows and sequence dependent setup times. *Computers & Industrial Engineering* 33: 773 – 776.

Kubiak, W., S. X. C. Lou, and Y. Wang. 1996. Mean flow time minimization in reentrant job shops with a hub. *Operations Research* 44: 764 – 776.

Lee, C. Y. and R. Uzsoy. 1999. Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research* 37: 219 – 236.

Mason, S. J. 2000. Minimizing Total Weighted Tardiness in Complex Job Shops. Ph.D. dissertation, Arizona State University.

Mason, S. J., J. W. Fowler, and W. M. Carlyle. 2001. A modified shifting bottleneck heuristic for minimizing the total weighted tardiness. In revision for *Journal of Scheduling*.

Nose, K., A. Hiramatsu, and M. Konishi. 1999. Using genetic algorithm for job-shop scheduling problems with reentrant product flows. *ETFA Proceedings of the 1999 7th IEEE International Conference on Emerging Technologies and Factory Automation* Vol 2.

Pinedo, M. and X. Chao. 1999. Operations Scheduling With Applications in Manufacturing and Services. Boston: Irwin/McGraw-Hill.

Roobeek, F. 1997. Better choice - a fuzzy logic based lot sequencing decision support system for operators in a job shop fab with reentrant process flows. *Proceedings of the 1997 IEEE International Symposium on Semiconductor on Manufacturing Conference*.

Smith, T. W. 2001. Semiconductor Equipment. *Standard & Poor's Industry Surveys* 69(5).

Uzsoy, R. 1995. Scheduling batch processing machines with incompatible job families. *International Journal of Production Research* 33: 2685 – 2708.

Semiconductor Industry Association reports global semiconductor market tops $200 billion mark for first time. *Semiseek News*. February 5, 2001. <http://www.semiseeknews.com/press_release2499.htm>, accessed July 5, 2001.

"Worldwide chip sales to fall 17% in 2001, says Dataquest," *Silicon Strategies* in *Semiconductor Business News*. May 8, 2001. <http://www.siliconstrategies.com/story/OEG20010508S0059>, accessed July 2, 2001.

AUTHOR BIOGRAPHIES

KASIN OEY is a Master's candidate in Industrial Engineering at the University of Arkansas. He received his B.S.I.E. at University of Wisconsin and will graduate in August 2001. His research interests include semiconductor

manufacturing, operations research, and production scheduling. His email address is <koey@uark.edu>

SCOTT J. MASON is an assistant professor in the Department of Industrial Engineering at the University of Arkansas. Dr. Mason received his B.S.M.E. and M.S.E. degrees from The University of Texas at Austin and his Ph.D. from Arizona State University. Prior to joining the University of Arkansas faculty, he worked for eight years in the semiconductor manufacturing industry, concentrating and manufacturing capacity analysis, future factory design, and global logistics issues. He is the Director of the Razorback Electronics Manufacturing Lab at the University of Arkansas. His research interests include modeling and analysis of complex manufacturing systems, applied operations research, and production planning and scheduling. His email address is <mason@uark.edu>.