Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

INTEGER PROGRAMMING-BASED REAL-TIME SCHEDULER IN SEMICONDUCTOR MANUFACTURING

Myoungsoo Ham
Industrial Engineering Department
Arizona State University
Tempe, AZ, USA

Young Hoon Lee
Dept. of Information and Industrial Engineering
Yonsei University
Seoul, South Korea

John W. Fowler
Industrial Engineering Department
Arizona State University
Tempe, AZ, USA

ABSTRACT

This paper demonstrates how an integer programming-based real-time scheduling heuristic approach can be applied for semiconductor manufacturing. Two integer programming formulations of a simplified version of this problem are proposed to model (1) a full-enumeration scheduling problem which minimizes the makespan (Cmax), and (2) a real-time scheduling problem which simply maximizes job assignments at the current state. The real-time scheduler's overall effectiveness in terms of solution quality and run time is evaluated through computer experiments. The real time scheduler is based on an iterative procedure to calculate the makespan, where a simulator is developed to read the integer programming output and to update the job and machine information at each state. The experimental study shows how a well-defined integer programming-based real-time scheduling heuristic can generate a near-optimal solution.

1 INTRODUCTION

With the skyrocketing cost of semiconductor equipment, companies are challenged to develop quick, robust, and efficient dispatching and scheduling systems. In most semiconductor manufacturing companies, the roles of the scheduling and dispatching systems are differentiated: the dispatcher typically calculates job-to-machine assignment only for a given set of jobs or machines. This limited scope enables the dispatcher to respond quickly (*i.e.*, within a few seconds) to system disturbances (Ham and Dillard 2005). In contrast, schedulers calculate a comprehensive series of job-to-machine assignments for all or a subset of jobs and machines available. The broader scope of scheduling may give a limitation on the scheduler's speed, agility, and overall robustness. Since semiconductor manufacturing is surrounded by unpredictable system disturbances at every second—including machine failures (down to up, up to down, and maintenance), lot hold/releases, and engineering changes--the scheduler's response is often obsolete. Of course, with minimal effort, re-scheduling can be employed to adjust the scheduler's decision. However, in many cases one disturbance forces the scheduler to generate a whole new set of schedules. To address this challenge, researchers have utilized improved computing power and algorithms to develop a more frequent short-term scheduling method that can run every 5 to 10 minutes (Bixby et al. 2006; Govind et al. 2007). Despite certainly a significant improvement, the solution is still quite far from a real-time decision solution, which is very much needed for the many unpredictable system disturbances in semiconductor manufacturing.

Several real-time scheduling approaches have been discussed in flexible manufacturing systems. Maley *et al.* (1988) proposed a methodology for optimizing the control of an automated manufacturing facility by utilizing real-time guidance from a historical knowledge-based artificial intelligence system. Yih and Thesen (1991) formulated a class of real-time scheduling problems as semi-Markov decision problems. Fowler *et al.* (1992) demonstrated a real time control of batch semiconductor manufacturing operations by using the knowledge of future lot arrivals. Kim and Kim (1994) proposed a scheduling mechanism where optimal job dispatching rule is dynamically selected based on information from discrete event simulation that evaluates a set of candidate rules.

Liao *et al.* (1996) developed an IP formulation to address the semiconductor scheduling problem. Because of the NP-hard computational complexity of the formulation, they used Lagrangian Relaxation theory by assuming that, with infinite machine capacity, there was no need to form batches. They further developed an iterative heuristic algorithm to adjust the dual solution to a near-optimal, feasible schedule by taking advantage of the marginal cost interpretation of Lagrange multipliers and the network structure of the flow balance equations.

More recently, Yu *et al.* (2002) discussed the scheduling of unrelated parallel machines in PWB manufacturing. They profiled a company that set out to create a schedule that would enable them to respond to unpredicted manufacturing events in under one minute. With speed and efficiency as their dual goals, the team constructed an IP formulation with Lagrangian Relaxation to absorb complicating constraints into the objective function.

Moving on to selected integer programming-based shop floor scheduling approaches, some of the more groundbreaking studies include Manne's (1960) linear programming model for a job-shop scheduling problem. Selen and Hott (1986) proposed a goal programming model of the flow-shop scheduling problem to consider multiple objectives. Liao and You (1992) then extended Manne's model by developing a job-shop scheduling problem with fewer constraints. Orcun *et al.* (2001) and Adams and Sherali (1990) presented several linearization techniques for the non linear integer programming models which contain quadratic cross-product terms. Stafford *et al.* (2005) investigated the performance of two families of integer programming models for solving the simple flow-shop problem to minimize makespan.

Other notable heuristic approaches for semiconductor scheduling include variants of the shifting bottleneck heuristic (Mason *et al*. 2002; Upasani *et al*. 2006), the generic algorithm (Malve and Uzsoy 2007; Chou 2007; Kashan *et al.* 2008), and simulated annealing (Erramilli and Mason 2006).

Practical integer programming-based approaches were first explored by Bixby *et al*. (2006). This paper described the practical implementation of a short interval area scheduler in a fully automated semiconductor fab. Their optimal solution contains a list of lot-step assignments to specific tools for a certain time horizon, and their algorithm uses dedicated schedulers for each fab process area to reduce computation time. Govind *et al*. (2007) presented a similar optimization-based scheduler in the photolithography area. They decomposed the fab scheduling problem into an individual area scheduling problem, enabling near real-time scheduling. In addition, they emphasized global targeting in order to compensate for the drawback of an area scheduler, which can lead to local optimality.

Building upon, expanding, and challenging studies past, this paper proposes an unprecedented method via integer programming-based real-time scheduling which generates a near-optimal solution in dynamic semiconductor manufacturing. Two integer programming models of a simplified version of this problem are presented for the scheduling problem in semiconductor manufacturing:

(1) a full-enumeration scheduling problem which minimizes the makespan (*Cmax)*,

(2) a real-time scheduling problem with production target that considers a line balance aspect while maximizing job assignments at the current state.

The integer programming based full-enumeration scheduling model generates the optimal *Cmax* in a single calculation. In contrast, the integer programming-based real time scheduler requires multiple iterations to calculate the final *Cmax*. At each iteration, the previously assigned jobs move to the next steps and the machines are released. The real time scheduler then calculates a schedule for the new state. The iterations continue until the last job finishes at the last step.

2 METHODOLOGY

We made several assumptions to simplify the semiconductor scheduling problem to make it solvable.

- No setups are required.
- All jobs are available at time zero.
- Each step has an identical processing time of one time unit.

Key among these assumptions is an assumption of identical processing time. There are two reasons : the first one is that we plan to conduct the computational study with a relatively large set of problem instances, *i.e.,* 30-60 jobs with 10-15 steps, in order to begin to understand the practical effect in a real fab. The assumption permits us to reduce the number of integer variables dramatically. The second reason is the software usability difficulty. Without the assumption, discrete event simulation software would be required. This is problematic because the software, which communicates with the linear programming solver at every job-to-machine assignment decision, is very difficult to implement. Our assumption permits us to write a small amount of code to simulate the wafer flow. To illustrate this, consider the very small problem instance defined in Table 1. Column 1 shows the sequence of steps, column 2 shows the processing time, column 3 shows the available machines, and column 4 shows the initial inventory positioned at the beginning of the specified step.

Table 1: Small process flow with three steps

Step	Processing Time	Machines	Jobs
1	1	1, 2	3
2	1	2	0
3	1	1,2	0

As shown, jobs are randomly queued at each step. Jobs at step *1* can be processed by either machine *1* or *2* with one unit of processing time. The jobs then move to step *2* where only machine *2* can be used. At step *3*, machines *1* and *2* are can be used. Finally, the jobs exit the flow. The objective is to minimize the maximum completion time of all jobs. The optimal *Cmax* of 5 is clearly demonstrated in Figure 1.

Job 1	S1, M1	S2, M2	S3, M1		
Job 2	S1, M2	Wait	S2, M2	S3, M1	
Job 3	Wait	S1, M1	Wait	S2, M2	S3, M1

t_0 t_1 t_2 t_3 t_4 t_5

Figure 1: Optimal schedule of small process flow with three steps

At time 0, the scheduler assigns job *1* at step *1* to machine *1* and job *2* at step *1* to machine *2*. At time *1*, the scheduler assigns job *1* at step *2* to machine *2* and job *3* at step *1* to machine *1*. At time *2*, it assigns job *1* at step *3* to machine *1* and job *2* at step *2* to machine *2*. At time *3*, it assigns job *2* at step *3* to machine *1* and job *3* at step *2* to machine *2*. Finally, at time *3*, the scheduler assigns job *3* at step *3* to machine *1*.

Figure 2 shows a non-optimal schedule with the optimal *Cmax* of 7.

Job 1	S1, M1	Wait	S2, M2	S3, M2			
Job 2	S1, M2	Wait	Wait	Wait	S2, M2	S3, M2	
Job 3	Wait	S1, M2	Wait	Wait	Wait	S2, M1	S3, M2

t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7

Figure 2: Non-optimal schedule of small process flow with three steps

At time *0*, the scheduler assigns job *1* at step *1* to machine *1* and job *2* at step *1* to machine *2*. At time *1*, the scheduler assigns job *3* at step *1* to machine *2* which makes machine *1* idle. At time 2, it assigns job *1* at step *2* to machine *2*. At time 3, it assigns job *1* at step *3* to machine *2*. At time 4, it assigns job *2* at step *2* to machine *2*. At time 5, it assigns job *2* at step *3* to machine *2* and job *3* at step *2* to machine *1*. Finally, at time 3, the scheduler assigns job *3* at step *3* to machine *2*.

With our manufacturing challenges clearly defined, we now turn our attention to the integer programming models, notes as follows:

Parameters:
t Number of steps (or states) ahead
I_{j0} Initial number of jobs at step j

Decision variables:
x_{ijk} 1 if machine i is assigned to step j at time k; *0 otherwise*

Bookkeepng variables:
I_{jk} Number of jobs at step j and time k
C_j Number of jobs already processed at step j

Note that index i refers to machines, index j refers to steps and index k refers to times. The problem description is addressed with an example. Section 2.1 and 2.2 show the integer programming models for the given problem.

2.1 Integer programming-based full-enumeration scheduling model (OPT)

$$\textit{Minimize} \quad Cmax \tag{1}$$

subject to

$$\sum_{i=1}^{m} x_{ijk} \le I_{jk} \qquad \forall j,k \tag{2}$$

$$\sum_{j=1}^{s} x_{ijk} \le 1 \qquad \forall i,k \tag{3}$$

$$I_{jk+1} = I_{jk} - \sum_{i=1}^{m} x_{ijk} + \sum_{i=1}^{m} x_{i,j-1,k} \quad (for\ j > 1) \quad \forall j,k \tag{4}$$

$$I_{jk+1} = I_{jk} - \sum_{i=1}^{m} x_{ijk} \quad (for\ j = 1) \quad \forall j,k \tag{5}$$

$$\sum_{i=1}^{m} \sum_{k=0}^{t} x_{ijk} = I_{j0} \quad \forall j \tag{6}$$

$$k \times x_{ijk} \leq Cmax \tag{7}$$

$$\forall\ x_{ijk}\ binary \tag{8}$$

In summary, the objective (1) is to minimize the maximum completion time of all jobs. Constraint set 2 ensures that the machine will not be assigned more inventory than available. Constraint set 3 ensures that the number of jobs assigned to a machine cannot exceed the machine capacity, which is 1. Flow balance is the focus of constraint sets 4 and 5, where inventory is defined as follows: Current inventory – Outgoing inventory + Incoming inventory.

Constraint set (6) ensures that each job visits each step. Constraint set 7 determines the makespan and Constraint set (8) imposes the binary restrictions on the *X* variables. This total enumeration model demonstrates the optimal solution that will minimize the maximum completion time of all jobs. Though we explore this model in theory, we recognize that it cannot actually be used in real semiconductor manufacturing, where every second is very expensive. Instead, we propose using an integer programming-based real-time scheduler model (discussed in section 2.2 below) to attack the practical problems in semiconductor scheduling. While the model below cannot guarantee global optimality, it can determine a solution in a near-real time.

2.2 Integer programming-based real-time scheduler model (RTS)

The objective of this model is to maximize job assignments at the current time bucket and the next *t* time buckets. As *t* is increased, the model generates a higher quality solution. When *t* is equal to *Cmax*, it becomes a full enumeration model. In this paper, we set *t* to 1 so that it considers the *current* and *very next* time buckets only. The model gives a higher weight to the current time bucket job assignment and a relatively lower weight to the next time bucket job assignment.

Do until $\forall I_{sk} = 0$

$$Maximize \quad \sum_{i=1}^{m} \sum_{j=1}^{s} \sum_{k=0}^{t} \left[\frac{1}{k+1} \right] x_{ijk} \times \frac{1}{C_j + 1} \tag{9}$$

Subject to

$$\sum_{i=1}^{m} x_{ijk} \leq I_{jk} \quad \forall j,k \tag{10}$$

$$\sum_{j=1}^{s} x_{ijk} \leq 1 \quad \forall i,k \tag{11}$$

$$I_{jk+1} = I_{jk} - \sum_{i=1}^{m} x_{ijk} + \sum_{i=1}^{m} x_{i,j-1,k} \quad (for\ j > 1) \tag{12}$$

$$I_{jk+1} = I_{jk} - \sum_{i=1}^{m} x_{ijk} \quad (for\ j = 1) \tag{13}$$

$$\forall\ x_{ijk}\ binary \tag{14}$$

$$I_{j0} = I_{j0} - \sum_{i=1}^{m} x_{ij0} + \sum_{i=1}^{m} x_{i,j-1,0} \quad (for\ j > 1) \tag{15}$$

$$I_{j0} = I_{j0} - \sum_{i=1}^{m} x_{ij0} \quad (for\ j = 1) \tag{16}$$

$$C_j = C_j + \sum_{i=1}^{m} x_{ij0} \qquad (17)$$

Loop

The real-time scheduler's constraint sets are similar to those of the full enumeration scheduler. Constraint set 10 prohibits the machines from being assigned more inventory than what is available. Constraint set 11 dictates that the number of jobs assigned to a machine should not exceed the machine capacity of 1. Constraint sets 12 and 13 ensure balance of flow. After the MIP with objective function (9) and constraint sets (10)-(14) has been run, equations 14 and the equations described in (15) – (17) are run to update the input parameters for the next iteration. The I_{j0} variables are the inventory positions and the weight variables (C_j) keep track of the number of job assignments at each step. The objective (9) gives a higher weight to an under-run step and a lower weight to an over-run step, which helps the jobs flow through the production line seamlessly.

3 COMPUTATIONAL STUDY

To evaluate the performance of our real-time scheduling heuristic approach, we conducted a broad set of computational experiments using randomly generated test instances. First, we used AMAT /RTD software language to generate the integer-programming models.

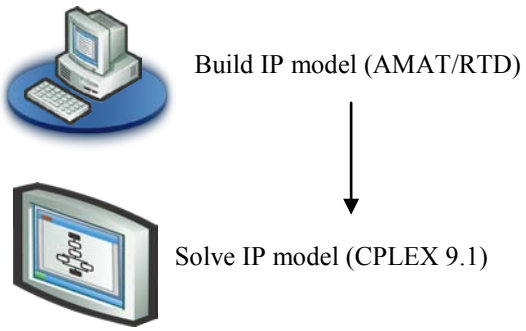

Figure 3: System flow of full-enumeration scheduling model

The optimization-based full-enumeration scheduling model (OPT) is then solved with ILOG/CPLEX 9.1 as shown in Figure 3, but the optimization based real-time scheduling model (RTS) is solved with the MIP solver within lpSolve 5.5 as shown in Figure 4. Solver lpSolve 5.5 has been selected because it provides a user-friendly interface. However, lpSolve 5.5 could not be used to solve the full enumeration model (OPT), which is quite sizable. Unlike the above full-enumeration scheduling model, the real-time scheduling model generates the decision only for a current time bucket. A simulator is therefore required to simulate the job flow. However, no simulator yet supports the flexible interface to the linear programming solver, especially when the linear programming model will be continuously updated based on a previous output. Since each job has an identical processing time, we developed a simulator to read the integer programming output and update the job information.

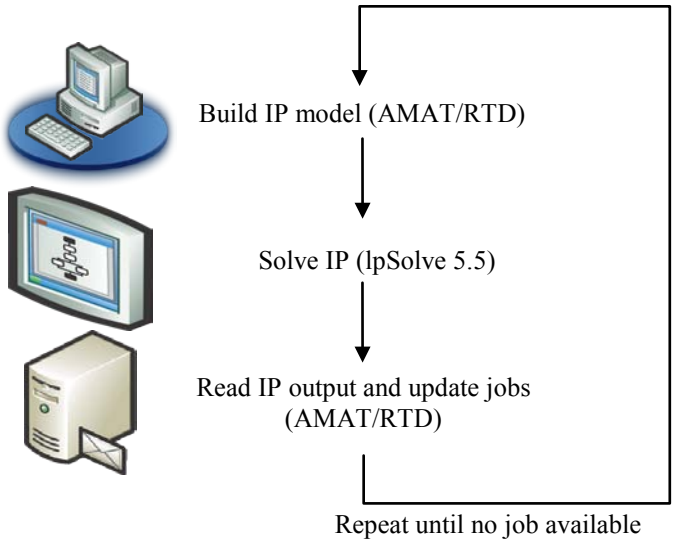

Figure 4: System flow of real time scheduling model

In this experimentation environment, section 3.1 demonstrates the computational results.

3.1 Small Dataset Results

For the small dataset, we randomly generated the numerical values of the problem parameters according to the previous assumptions. Table 2 presents the 2*2*2*2combinations of the factors to be investigated on the semiconductor manufacturing flow problem instances (Montgomery 2004).

Table 2: Factors for the small problem instances

Factor	Levels	Total levels
Number of steps	10, 15	2
Number of jobs	30, 60	2
Number of machines	3, 5	2
Scheduling Methods	OPT, RTS	2
Replication	Randomly generated machine qualifications	5
Total Problem Instances	2*2*2*2*5	80

The results in Table 3 reveal the Cmax values for the problem instances.. Column 1 shows the number of steps, column 2 shows the number of machines, and column 3 shows the replications. Column 4 reports the best Cmax from OPT model generated by CPLEX, column 5 reports the best Cmax from RTS model, and column 6 reports the percentage above optimal which is calculated as Cmax (RTS) / Cmax(OPT). The results for 60 jobs instances haves the same columns. As one can see, RTS maintains a clear focus on long-term performance. It does so by giving a higher weight to an under-run step and a lower weight to an over-run step, enabling jobs to flow through the production line more seamlessly. This approach results in 1%-3% above optimal (on average).

Table 3: Results (*Cmax*) for small problem instances

			30 jobs			60 jobs		
		Run	OPT	RTS	RTS/OPT	OPT	RTS	RTS/OPT
10 steps	3 machines	1	55	56	1.02	125	129	1.03
		2	92	92	1.00	161	161	1.00
		3	41	43	1.05	122	124	1.02
		4	36	36	1.00	99	100	1.01
		5	75	76	1.01	143	145	1.01
	sub avg				1.02			1.01
	5 machines	1	17	18	1.06	56	57	1.02
		2	24	24	1.00	72	72	1.00
		3	29	29	1.00	63	64	1.02
		4	27	29	1.07	65	68	1.05
		5	35	36	1.03	64	64	1.00
	sub avg				1.03			1.02
15 steps	3 machines	1	52	53	1.02	124	125	1.01
		2	77	77	1.00	165	166	1.01
		3	65	65	1.00	127	127	1.00
		4	62	62	1.00	140	141	1.01
		5	69	70	1.01	173	174	1.01
	sub avg				1.01			1.01
	5 machines	1	37	38	1.03	85	86	1.01
		2	52	54	1.04	64	65	1.02
		3	42	44	1.05	88	90	1.02
		4	38	38	1.00	89	90	1.01
		5	37	40	1.08	71	72	1.01
	sub avg				1.04			1.02
	grand avg				1.02			1.01

Regarding the computational run time shown in Table 4, RTS returned the solutions within one second. Although the size of the problem instances tested is small, we expect that the RTS solution time will be reasonable even for large problem instances since the problem is solved for only a single period at a time Notice that the full enumeration IP model (OPT) requires setting a maximum value of *t*. Although the *Cmax* value generated from RTS is used as *t*, which provides a very strong lower bound (permitting us to reduce the number of integer variables), the integer programming-based full enumeration model still takes up to 3404 seconds.

Table 4: Solution time in seconds of small problem instances

		Run	30 jobs	60 jobs
10 steps	3 machine	1	2.08	17.67
		2	3.70	4.32
		3	1.68	23.18
		4	0.77	13.20
		5	6.60	35.22
	sub avg		2.97	18.72
	5machines	1	0.40	7.63
		2	0.17	4.87
		3	1.55	3.48
		4	1.72	5.17
		5	2.52	4.80
	sub avg		1.27	5.19
15 steps	3machines	1	12.92	299.97
		2	21.47	1963.10
		3	17.08	71.13
		4	8.92	598.13
		5	24.58	3404.35
	sub avg		16.99	1267.34
	5machines	1	6.92	14.83
		2	22.57	9.05
		3	21.32	70.93
		4	6.02	41.18
		5	6.63	22.48
	sub avg		12.69	31.69
grand avg			8.48	330.73

Table 5 represents the ANOVA analysis of the four factors in the experimental design. Based on the p-values, we see that the model is a significant factor in this design of experiment.

Table 5: ANOVA results of small dataset

Source	DF	Sum of Squares	F Ratio	Prob > F
Machines	1	73804.8	177.5448	<.0001
Steps	1	1657.63	3.9876	0.0482
Jobs	1	109565.63	263.571	<.0001
Scheduling Methods	2	4615.8	5.5519	0.005

4 CONCLUSION

In this paper, we have proposed a binary inter programming-based real-time scheduling heuristic approach for non-standard job-shop flow. To accomplish this, we developed two integer programming formulations of a simplified version of this problem, as follows: (1) a full-enumeration scheduling problem which minimizes *Cmax* and (2) a real-time scheduling problem with a production target that considers a line balance aspect while maximizing job assignments at the current state.

The computational study shows that the well-defined integer programming based real-time scheduling model can generate a near-optimal solution, 1-3% above optimal in average. We will later try more realistic problem instances with 25 steps, 120 jobs, and 7 machines.

A real-time scheduling approach is even more important in a dynamic fab environment where a diverse range of unpredictable system disturbances are the norm.

For future research, an integer programming-based real-time semiconductor scheduler--with batching, setups, delay times, and hot job--will be developed. We note that a real-time scheduling approach similar to the one developed in this paper might be appropriate for other industries, including airline scheduling and chemical blending scheduling.

REFERENCES

Adams, W. P., and H. D. Sherali. 1990. Linearization strategies for a class of zero-one mixed integer programming problems. *Operations Research*, 38(2): 217-226.

Bixby, R., R. Burda, and D. Miller. 2006. Short-interval detailed production scheduling in 300mm semiconductor manufacturing using mixed integer and constraint programming. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 148-154.

Chou, F. D. 2007. A joint GA+DP approach for single burn-in oven scheduling problems with makespan criterion. *The International Journal of Advanced Manufacturing Technology*, *35*(5).

Erramilli, V. and S. J. Mason. 2006. Multiple orders per job compatible batch scheduling. *IEEE Transactions on Electronics Packaging Manufacturing*, 29(4): 285-296.

Fowler, J. W., G. L. Hogg, and D. T. Phillips. 1992. Real-time control of multiproduct bulk service diffusion/oxidation processes. *IIE Transactions*, 24: 84–96.

Govind, N., E. Bullock, L. He, B. Iyer, M. Krishna, and C. Lockwoood. 2007. Integrated targeting, near real-time scheduling, and dispatching with automated execution in semiconductor manufacturing. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 87-92.

Ham, M. and F. Dillard. 2005. Dynamic photo stepper dispatching/scheduling in wafer fabrication. *IEEE International Symposium on Semiconductor Manufacturing*, 75-79.

Kashana, A. H.., B. Karimi, and M. Jenabi. 2008. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Computers & Operations Research*, 35(4): 1084-1098.

Kim, M.-H., and Y.-D. Kim. 1994. Simulation-based real-time scheduling mechanism in a flexible manufacturing system. *International Journal of Production Research*, 13: 85–93.

Liao, C. J., and C. T. You. 1992. An improved formulation for the job-shop scheduling problem. *Journal of Operation Research Society*, *43*(11): 1047-1054.

Liao, D. Y., S. C. Chang, K. W. Pei, and C. M. Chang. 1996. Daily scheduling for R&D semiconductor fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 9(4): 550-561.

Maley, J. G., S. Ruiz-Meir, and J. J. Solberg. 1988. Dynamic control in automated manufacturing: a knowledge-integrated approach. *International Journal of Production Research*, 26: 1739–1748.

Malve, S., and R. Uzsoy. 2007. A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research*, 34(10): 3016-3028.

Manne, A. S., 1960. On the job-shop scheduling problem. *Operations Research*, *8*(2): 219-223.

Mason, S. J., J. W. Folwer, and W. M. Carlyle. 2002. A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5(3): 247-262.

Montgomery, D. C., 2004. *Design and Analysis of Experiments*, 6th Edition, New York: Wiley.

Orcun, S., I. K. Altinel, and O. Hortacsu. 2001. General continuous time models for production planning and scheduling of batch processing plants: mixed integer linear program formulations and computational issues. *Computers and Chemical Engineering*, 25: 371-389.

Selen, W., and D. Hott. 1986. A mixed-integer goal-programming formulation of the standard flow-shop scheduling problem. *Journal of the Operational Research Society*, 37(12): 1121-1128.

Stafford, E., F. Tseng, and J. Gupta. 2005. Comparative evaluation of MILP flowshop models. *Journal of the Operational Research Society*, 56: 88-101.

Upasani, A. A., R. Uzsoy, and K. Sourirajan. 2006. A problem reduction approach for scheduling semiconductor wafer fabrication facilities. *IEEE Transactions on Semiconductor Manufacturing*, 19(2): 216-225.

Yih, Y., and A. Thesen. 1991. Semi-Markov decision models for real-time scheduling. *International Journal of Production Research*, 29: 2331–2346.

Yu, L., H. Shih, M. Pfund, M. Carlyle, and J. W. Fowler. 2002. Scheduling of unrelated parallel machines: an application to PWB manufacturing. *IIE Transactions*, 34(11): 921-931.

AUTHOR BIOGRAPHIES

MYOUNGSOO HAM received Ph.D. degree in Industrial Engineering from Arizona State University in 2008, and M.S. degree in operations research and industrial engineering from the University of Texas at Austin. He is currently working as a member of technical staff at Samsung Austin Semiconductor. Prior to his current position, Ham worked for Samsung Austin Semiconductor as a scheduling/dispatching manager, ILOG's Semiconductor Business Division as Senior Product Manager, Texas Instruments as IE Manager, AMD as Senior IE engineer, and Samsung Korea Semiconductor as Automation Engineer. His research interests include modeling, dispatching & scheduling of manufacturing. His email is <hamcruise@gmail.com>.

YOUNG HOON LEE received M.S and Ph.D. degrees in Industrial Engineering from Columbia University in 1989 and 1992, respectively. He is currently working as an Associate Professor at Yonsei University, Department of Information and Industrial Engineering, Seoul, Korea. Prior to his current position, Lee worked for Samsung Electronics, Semiconductor Division at Kiheung, Korea, as a senior manager of System Integration team. His research interest includes modeling and analysis of semiconductor manufacturing for the planning, scheduling and the supply chain management. His email is <youngh@yonsei.ac.kr >.

JOHN W. FOWLER is the Avnet Professor of Supply Networks and a Professor of Industrial Engineering at Arizona State University (ASU). His research interests include modeling, analysis, and control of manufacturing and service systems. His research has been supported by the National Science Foundation, Semiconductor Research Corp., International SEMATECH, Asyst, IBM, Intel, Motorola, Infineon Technologies, and ST Microelectronics. Dr. Fowler is an author on over 70 journal publications, 100 conference papers, and 10 book chapters. He is the founding editor on the new journal *IIE Transactions on Healthcare Systems Engineering*. He is also an Area Editor for *SIMULATION: Transactions of the Society for Modeling and Simulation International* and for *Computers and Industrial Engineering*, an Associate Editor for *IEEE Transactions on Semiconductor Manufacturing*, and on the Editorial Board for the *Journal of Simulation*. He is a Fellow of the Institute of Industrial Engineers, the INFORMS Vice President for Chapters/Fora, and is on the Winter Simulation Conference Board of Directors. His email address is <john.fowler@asu.edu>.