Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

SIMULATION-BASED ASSESSMENT OF BATCHING HEURISTICS IN SEMICONDUCTOR MANUFACTURING

Lars Mönch
Ilka Habenicht

Institute of Information Systems
Technical University of Ilmenau
Helmholtzplatz 3, P.O. BOX 100565
D- 98684 Ilmenau, GERMANY

ABSTRACT

In this paper, we investigate the performance of different dispatching and scheduling heuristics for batching tools in a semiconductor wafer fabrication facility (wafer fab) by means of discrete event simulation. Because the processing times of lots on batching tools are quite large compared to those of other processes, careful batching decisions may have a great impact on the performance of the entire wafer fab. In a first step, we investigate the performance of certain modifications of the Apparent Tardiness Cost (ATC) dispatching rule that do not take into account future lot arrivals. In a second step, we extend this approach by considering future lot arrivals. In a last step, we combine a genetic algorithm for assignment of the batches to parallel machines with the ATC rule, which takes future lot arrivals into account. We present results of simulation experiments with the different heuristics.

1 INTRODUCTION

The manufacturing of integrated circuits (IC) on silicon wafers is a complex production process (cf. Atherton and Atherton 1995, Uzsoy et al. 1992, Schömig and Fowler 2000). Between 250-500 process steps on 50-120 different types of equipment are required to produce a medium complexity circuit. A mix of different process types, i.e. batch processes and single wafer processes, sequence-dependent setup times, very expensive equipment and re-entrant flows are typical for this type of production. The production process of circuits requires an extremely clean manufacturing area, i.e. usually the production of circuits takes place in clean room environments. The production of Application Specific Integrated Circuits (ASIC) is additionally characterized by a wide range of product types and the demand to achieve good delivery performance.

A single product moves through the wafer fab in lots. Each lot consists of several wafers. It is possible to place up to 800 circuits on a single wafer. The circuits are made up of layers. Because of the layered nature of the circuits it is possible to split up the reentrant parts of the production flows into the following groups: cleaning, metal deposition, chemical vapor deposition (CVD), photolithography, etching and ion implantation, and inspection/control steps.

Based on this grouping, a decomposition of the entire factory in individual work areas takes place. Unfortunately, the groups obtained in the decomposition are not always the same for different layers. The recursive nature of the process flows is one of the main sources of difficulty in planning, scheduling and controlling wafer fabs.

In this paper, we study different dispatching and scheduling strategies for a wafer fab. We start by using versions of the Apparent Tardiness Cost Dispatching (ATC) rule that do not take information on future lot arrivals into account. Then we consider modifications of the same dispatching rule that take future lot arrivals into account. This type of information is usually provided by the manufacturing execution system (MES) on the shop-floor. We use a time window technique in order to form batches. As a third strategy, we use a genetic algorithm in order to assign batches to parallel machines. Here again, we consider future lot arrivals in the decision-making process.

We use a simulation model of a full wafer fab in order to determine the performance of our three different algorithms. While the case of cycle-time-related performance measures has been investigated quit well we do not know much on due-date-oriented performance measures.

The paper is organized as follows. In the next section, we give a detailed description of the problem under consideration and summarize some related research. The main part of the paper deals with the presentation of the different dispatching and scheduling heuristics. Then we continue with describing the simulation test-bed used in the experiments. We present the results of computational experiments in section 5.

2 PROBLEM DESCRIPTION

2.1 Batching Issues

Batching is an important issue in semiconductor manufacturing. Because the processing times of the lots on the batching tools are usually very long compared to other processes, batching decisions may affect the performance of the entire wafer fab. In batch operations, even though several lots can be processed at the same time, lots of different families cannot be processed together due to the chemical nature of the process. Lots that can be processed together form one family.

Furthermore, depending on customer requirements, some lots have a higher priority index than others and products need to meet internal/external due dates. In the presence of unequal ready times of the lots it is sometimes advantageous to form a non-full batch, in other situations it is a better strategy to wait for next job arrivals in order to increase the fullness of the batch, i.e., allow for delayed schedules.

2.2 Related Literature for Batching Problems

Batching problems in wafer fabs have been addressed over the last decade by many authors and are still a challenging issue.

We refer to the work done by Fowler et al. (1992) and (2001). The authors propose several heuristics for real-time control for both single product and multi-product batch tools in order to minimize the flow-time of the lots. Other heuristics related to this issue are suggested by Glassey and Resende (1991). Future lot arrivals are considered during the decision-making process.

In (Robinson et al. 1995) the authors suggest the use of both down-stream and up-stream information for batching decisions.

Rolling horizon and time window approaches were suggested by Chand et al. (1997) but without considering batching problems.

In a more recent work Cigolini et al. (2002) propose a new look-ahead heuristic for minimizing cycle time and guarantying a certain throughput-level.

Static batching problems with incompatible families and parallel machines are studied by Mönch et al. (2002). Different genetic algorithms are proposed for the assignment of lots or batches to machines.

Several batching heuristics are investigated by Akçali et al. (2000) by using discrete event simulation.

However, only little is known on batching heuristics for minimizing due-date-oriented performance measures like total weighted tardiness or number of late lots on an entire wafer fab level.

2.3 Notation

We fix one batch tool group. We have to make a batching decision, i.e. we have to decide whether we leave a certain tool idle in order to wait for future lot arrivals to form a batch or to form a batch only from the set of lots waiting in front of the tool group for processing.

The following notation is used throughout the rest of the paper.

1. Lots fall into different incompatible families that cannot be processed together. There exist F such families.
2. There are n lots that have to be scheduled.
3. The fixed tool group contains m identical machines in parallel.
4. There are n_j lots of family j to be used for forming and sequencing of batches: $\sum_{j=1}^{F} n_j = n$.
5. Lot i of family j is represented as ij
6. The priority weight for lot i of family j is represented as w_{ij}.
7. The due date (with respect to the batching tool group) of lot i of family j is represented as d_{ij}.
8. The processing time of lots in family j is represented as p_j.
9. The ready time of lot i in family j is represented as r_{ij}.

We use the notation $x^+ := max(x,0)$ for abbreviation. In this research, we frequently refer to the total weighted tardiness measure for a given schedule. This performance measure is defined as follows:

$$TWT := \sum w_{ij} \left(c_{ij} - d_{ij}\right)^+ . \qquad (1)$$

3 BATCHING HEURISTICS

In this section, we describe three different batching heuristics. We are interested in the question how much we can gain from using more sophisticated (and, of course, computationally more expensive) heuristics for making batching decisions.

3.1 Static Batch Dispatching Heuristic (SBDH)

The first heuristic uses a modification of the well-known Apparent Tardiness Cost Dispatching Rule (Vepsalainen and Morton 1987). The rule was adapted by Mason et al. (2002) to the scheduling of batch machines. The ATC in-

dex $I_{ij,stat}(t)$ for job i belonging to family j calculated at time t is given below:

$$I_{ij,stat}(t) = \frac{w_{ij}}{p_j} exp\left(-\frac{(d_{ij} - p_j - t)^+}{k\bar{p}}\right). \quad (2)$$

Here, we denote by k a scaling parameter and by $\bar{p}$ the average processing time of the remaining jobs. In order to form the batches, we sequence the lots waiting in front of the batch tool group in descending order with respect to their $I_{ij,stat}(t)$ index. We take the first B lots of this sequence in order to form the batch that has to be processed next on the tool available for processing. We repeat the calculations several times for different values of k in order to find an optimal k parameter with respect to the total weighted tardiness value for the lots waiting for processing.

This strategy is, of course, a full-size batch strategy. It does not take any future lot arrivals into account. It is known that the rule performs well in a static parallel batch machine environment with respect to minimizing total weighted tardiness (Mönch et al. 2002). However, we expect a less good performance in a dynamic environment.

3.2 Dynamic Batch Dispatching Heuristic (DBDH)

The second heuristic takes future lot arrivals into account. For that purpose, we consider a time window $(t, t+\Delta t)$. We denote by

$$M(i,t,\Delta t) := \{ij \mid r_{ij} \le t + \Delta t\} \quad (3)$$

the set of lots that are ready for processing at time t or will arrive inside the window. Usually, we choose a certain portion of the average processing time of the waiting lots as Δt.

We sort the elements of $M(i,t,\Delta t)$ with respect to the index

$$I_{ij,dyn}(t) = \frac{w_{ij}}{p_j} exp\left(-\frac{(d_{ij} - p_j - t + (r_{ij} - t)^+)^+}{k\bar{p}}\right) \quad (4)$$

in a descending order. In a next step, only the first $\#lots$ lots of $M(i,t,\Delta t)$ maximum are considered by the heuristic. Here, we denote by $\#lots$ a fixed number that is a parameter of the heuristic. We build all batch combinations using these lots.

For each formed potential batch we calculate the batch index

$$I_{bj1}(t) = \frac{w_{bj}}{p_j} \exp\left(-\frac{(sl + rt)^+}{k\bar{p}}\right) \min\left(\frac{n_{bj}}{B}, 1\right), \quad (5)$$

where we denote by

$d_{bj} := \min_{j \in B_{bj}}(d_{ij})$: minimum due date among the jobs contained in the batch,

$r_{bj} := \max_{j \in B_{bj}}(r_{ij})$: maximum ready time of the jobs in the batch,

w_{bj} : average weight of the lots that form the batch,

n_{bj} : number of lots in the batch.

Here, we use for abbreviation $sl := d_{bj} - p_j - t$ and $rt := (r_{bj} - t)^+$ in (5).

This rule does not necessarily form full batches, however, it is sometime more advantageous to leave a tool idle and wait for an important lot instead of processing a full batch with unimportant lots.

3.3 Genetic Algorithm-Based Batching Heuristic (GABH)

We consider a genetic algorithm for assigning formed batches to machines of the group of parallel batch machines. We use the DBDH heuristic from the last section in order to form batches.

The genetic algorithm maintains a population of assignments of batches to machines. In this research, we consider the batch-based representation used by Mönch et al. (2002). After the assignment step we have to sequence the batches on each single machine. In order to carry out this step, we again consider the batch index (5).

After the sequencing phase, it is possible to calculate the total weighted tardiness values on each single machine. By aggregating these values, we obtain the total weighted tardiness value for the entire schedule. We use this value for an evaluation of the schedule by a fitness function. The genetic algorithm changes its population by using crossover and mutation operations. After a certain number of iterations (called generation) the genetic algorithm converges to a good solution with respect to total weighted tardiness.

We use classes from the GaLib library (Wall 1999) written in the C++ programming language in order to implement the genetic algorithm.

GABH is a scheduling heuristic that simultaneously keeps track of the entire tool group, whereas SBDH and DBDH are to a certain extend only dispatching heuristics. We use GABH in a rolling horizon manner.

4 FRAMEWORK FOR EXPERIMENTATION

We use a discrete-event simulation tool and a simulation model of a wafer fab to evaluate the performance of SBDH, DBDH, and GABH. This approach is widely used in the research community to estimate the performance of new algorithms (cf. Horiguchi et al. 2001, Habenicht and Mönch 2002).

We use the basic architecture described by Mönch et al. (2002) including the simulation tool AutoSched AP 7.1. This architecture centers around a data storage (called data model) that contains all the information required to run more sophisticated dispatching and scheduling algorithms. We extend the data model by additional classes and attributes to make it usable for the three algorithms. The basic architecture can be seen in Figure 1.

Because the simulation tool includes the AP Framework, which is written in the C++ programming language, the integration of the GALib library used for implementing GABH only requires little effort. The used framework is shown in Figure 1.

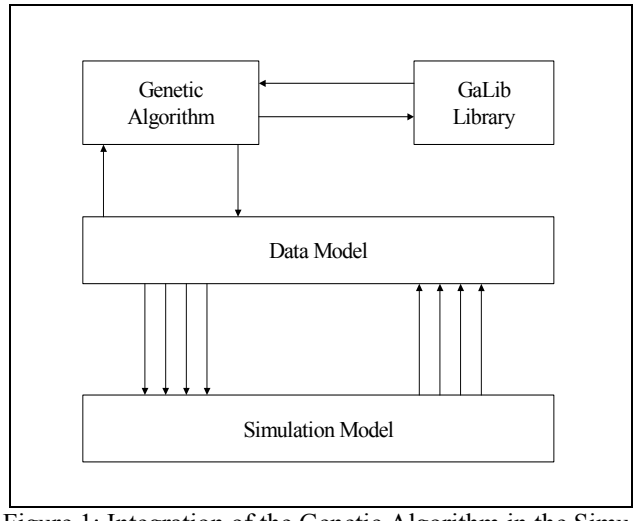

Figure 1: Integration of the Genetic Algorithm in the Simulation Environment

We implement two additional dispatching rules according to SBDH and DBDH.

We use the MIMAC test data set 1 (Fowler and Robinson 1995) in a slightly modified version. This simulation model consists of two different process flows (A,B) with 200 process steps and over 80 different tool groups.

There are 16 batching tool groups among the tool groups of the MIMAC test data set 1. Tool group OXIDE_1 is bottleneck of the wafer fab. Table 1 provides information on this particular batching tool group. In Table 1, we denote by B_{min} the minimum batch size in lots, by B_{max} the maximum batch size in lots. We denote the minimum processing time (measured in minutes) by p_{min}, the corresponding maximum processing time (in minutes) is denoted by p_{max}. The given utilization was determined by simulation experiments with the First In First Out (FIFO) dispatching rule. We decided to apply the SBDH, DBDH and GABH rule to this tool group.

Table1: Bottleneck Batching Tool Group Information

Tool Group	# tools	B_{min}	B_{max}	p_{min}	p_{max}	Utilization [%]
OXIDE_1	3	2	6	135	1410	84.19

We use a (process step) slack-based dispatching rule for the non-batching tools. The rule selects the lot with the smallest slack for that process step. For the calculation of the slack of the lots waiting in front of a certain machine, we simply multiply the processing time with a dynamic flow factor. For that purpose we calculate the difference between the due date of the lot and the current time. Based on this information, we assign a flow factor to each lot (cf. Habenicht and Mönch 2002 for more details on this infinite capacity approach). This scheme allows to determine end dates for the single process steps, i.e. future lot arrival information are available at the batch tools. We repeat the calculation of the flow factors every 15 minutes.

We expect that better estimates through avoiding the infinite capacity approach and using a finite capacity algorithm (cf. Habenicht and Mönch 2002 for such an algorithm) instead. However, carrying out the details is part of future research.

In our experiments, we use a moderate workload of the system. Machine failures are exponentially distributed. We use a forecast horizon of three months. The model is initialized by using a work in process (WIP) distribution of the wafer fab. The length of a single simulation run was 100 days in our experiments. We take five independent replications of each simulation run in order to obtain statistically significant results.

5 EXPERIMENTAL DESIGN AND COMPUTATIONAL RESULTS

In this section, we present the results of computational experiments. Therefore, we first define performance measures, then we describe a factorial design. We finish the section with a presentation of computational results.

5.1 Used Performance Measures

The following performance measures are used:

- Total weighted tardiness of the lots that are released and finished within the planning horizon

under consideration. We define the weighted tardiness of lot j as follows:

$$T_j := w_j \left(C_j^r - d_j\right)^+, \tag{6}$$

where we denote by C_j^r the realized completion time, by d_j the due-date, and by w_j the weight of lot j. In order to derive the performance measure we sum up the corresponding T_j for all lots. We denote this quantity by TWT_{total}.

- Average cycle time of product P: $CT(P)$.
- Throughput of the wafer fab (number of completed wafers): *TP*.

5.2 Design of Experiments

We are interested in studying the behavior of the batching heuristics under different system conditions. We identify the following factors that might influence the performance of the investigated heuristics. We distinguish between factors that characterize the manufacturing system and factors that are used for parameter setting in the heuristics. The following factors belong to the first group:

- number of families,
- due date setting,
- weight setting.

We consider the following factors in the second group:

- length of the time window,
- maximum number of lots used for considering all batch combinations,
- setting of the look-ahead parameter k.

In our experiments, we fix a heuristic and we only consider the case of optimal k value setting (with respect to TWT). The maximum number of lots used for considering all batch combinations is ten. Likewise, we only consider the case of two different product flows. We use the factorial design given in Table 2.

Other interesting factors are the workload of the fab and the used strategy for order release (daily vs. weekly). However, in this study we fix these factors. Another factor of interest is the update frequency of the infinite capacity algorithm that enables us to obtain new estimates for future lot arrivals.

Using DHBH and GABH, we derive a new schedule for the tool group every time a new lot arrives. The remaining lots of the batch are chosen from the waiting lots of the same family.

5.3 Results of Computational Experiments

In this section, we present the results of computational experiments with the suggested heuristics. The resulting performance measures are presented in terms of the ratio of the performance measure value of the heuristic and the FIFO rule performance measure value.

Table 2: Used Factorial Design

Factor	**Level**	
	I	**II**
Due Date	Flow Factor $f := 2.0$	Flow Factor $f := 2.0$ for 50% of the lots, $f := 1.5$ for 50% of the lots
Weight	With probability $p_1 = 0.50$ $w_j = 1$, with probability $p_2 = 0.35$ $w_j = 5$, with probability $p_3 = 0.15$ $w_j = 10$	With probability $p_1 = 0.50$ $w_j = 1$, with probability $p_2 = 0.45$ $w_j = 2$, with probability $p_3 = 0.05$ $w_j = 10$
Time Window Size	25 % of the average processing time of the queuing lots in front of the batching tools	50 % of the average processing time of the queuing lots in front of the batching tools

Because we do not take future lot arrivals into account for SBDH, we have to consider only the first and the second factor from Table 2. Therefore, we have to perform eight different experiments. We use the triple (dispatching rule, level from factor 1, level from factor 2) in order to indicate the used factor combination.

The results of the SBDH-based batching strategy are shown in Table 3.

Table 3: Results for SBDH

Factor Combination	TWT_{total}	*CT*	*TP*
1 FIFO (I-I)	1.0000	1.0000	1.0000
2 SBDH (I-I)	0.1662	0.9721	1.0133
3 FIFO (I-II)	1.0000	1.0000	1.0000
4 SBDH (I-II)	0.2188	0.9748	1.0113
5 FIFO (II-I)	1.0000	1.0000	1.0000
6 SBDH (II-I)	0.2600	0.9600	1.0100
7 FIFO (II-II)	1.0000	1.0000	1.0000
8 SBDH (II-II)	0.3900	0.9600	1.0200

The corresponding results of the DBDH-based batching strategy can be obtained from Table 4. Here, we have to perform totally twelve different experiments because the third factor (time window size) is also important. Consequently, we use the notation (dispatching rule, level from

Table 4: Results for DBDH

Factor Combination	TWT_{total}	CT	TP
1 FIFO (I-I)	1.0000	1.0000	1.0000
2 DBDH (I-I-I)	0.1031	0.9685	1.0099
3 DBDH (I-I-II)	0.1191	0.9725	1.0089
4 FIFO (I-II)	1.0000	1.0000	1.0000
5 DBDH (I-II-I)	0.0918	0.9677	1.0135
6 DBDH (I-II-II)	0.1475	0.9732	1.0056
7 FIFO (II-I)	1.0000	1.0000	1.0000
8 DBDH (II-I-I)	0.2913	0.9631	1.0081
9 DBDH (II-I-II)	0.3035	0.9703	1.0082
10 FIFO (II-II)	1.0000	1.0000	1.0000
11 DBDH (II-II-I)	0.4230	0.9588	1.0089
12 DBDH (II-II-II)	0.4363	0.9643	1.0085

factor 1, level from factor 2, level from factor 3) in order to indicate the investigated factor combination.

The corresponding results of GABH are presented in Table 5. Similar to the previous case, we have to consider twelve different factor combinations. The used notation is the same as in Table 4.

Table 5: Results for GABH

Factor Combination	TWT_{total}	CT	TP
1 FIFO (I-I)	1.0000	1.0000	1.0000
2 GABH (I-I-I)	0.1107	0.9651	1.0117
3 GABH (I-I-II)	0.0950	0.9680	1.0100
4 FIFO (I-II)	1.0000	1.0000	1.0000
5 GABH (I-II-I)	0.1163	0.9710	1.0084
6 GABH (I-II-II)	0.1498	0.9775	1.0097
7 FIFO (II-I)	1.0000	1.0000	1.0000
8 GABH (II-I-I)	0.2500	0.9572	1.0094
9 GABH (II-I-II)	0.3085	0.9728	1.0085
10 FIFO (II-II)	1.0000	1.0000	1.0000
11 GABH (II-II-I)	0.4137	0.9589	1.0127
12 GABH (II-II-II)	0.4796	0.9726	1.0090

From the performed experiments, we can verify that all three batching heuristics outperform the FIFO rule. The results of the total weighted tardiness are sensitive to factor settings. The consideration of future lot arrivals leads to an reduction of the total weighted tardiness. The results of DHBH and GABH differ only slightly. This is caused by the fact that the number of batches that are assigned to the parallel machines via the genetic algorithm is small. Hence, the room for improvements is also small. GABH performs better than DHBH only for a large numbers of batches.

The improvement of the total weighted tardiness value obtained by using any of the three heuristics is sensitive to due date and weight setting. The total weighted tardiness of the factor combinations I-II, II-I, and II-II is reduced compared to factor combination I-I. In the case of level II for the due-date, half of the lots have a tight due date. Therefore, it is rather hard for any of the BATC type dispatching rules to differentiate between the high prioritized lots. Hence, a huge portion of late lots have to wait in front of the batch tools.

In Table 6 we summarize the results for factor combination I-I and different time window sizes Δ. It is possible to come up with a time window size with a minimal total weighted tardiness value.

Table 6: Results for Different Time Window Settings

Heuristic/ Time Window (percent of average processing time of the queuing lots)	TWT_{total}	CT	TP
1 FIFO	1.0000	1.0000	1.0000
2 SBDH	0.1619	0.9721	1.0133
3 DBDH			
3.1 25%	0.1031	0.9685	1.0099
3.2 50%	0.1191	0.9725	1.0089
3.3 75%	0.1384	0.9761	1.0049
4 GABH			
4.1 25%	0.1107	0.9651	1.0117
4.2 50%	0.0950	0.9680	1.0100
4.3 75%	0.2506	1.0001	1.0043

A larger time window size Δ leads to more uncertain lot arrival time predictions. The accuracy of the lot arrival time prediction is an important precondition for high-quality results of DBDH and GABH, because these strategies are based on future lot arrival information. This becomes clear when considering batch utilization (average number of lots that form a batch), the overall utilization of the tools and average queue-size in front of the batching tool groups. The utilization data are shown in Table 7. Choosing a larger time window size Δ leads to a larger

Table 7: Utilization of the Batching Tool Group (OXIDE_1)

Heuristic/ Time Window (percent of average processing time of the queuing lots)	Batch Utilization	Utilization	Average Queue-Size
1 FIFO	1.0000	1.0000	1.0000
2 SBDH	0.9003	1.0107	0.8922
3 DBDH			
3.1 25%	1.0595	0.8640	0.9105
3.2 50%	1.0436	0.8426	1.0431
3.3 75%	1.0499	0.8028	1.4321
4 GABH			
4.1 25%	1.0737	0.8611	0.9793
4.2 50%	1.0771	0.8251	1.0528
4.3 75%	1.2705	0.7405	2.8811

batch utilization and a larger queue size. The tools have to wait longer for lots that arrive during the given time window. Therefore, the utilization of the tools decreases. If the prediction of lot arrival times is incorrect, the idle times of the tool waiting for lot arrivals increase and other lot with lower priority have to wait.

The FIFO dispatching rule shows an interesting behavior. The batch utilization here is larger than the batch utilization of the SBDH strategy. The reason for this behavior lies in the large queue-size and the large utilization caused by the FIFO dispatching regime. The large number of lots in the queues leads to a higher utilization of batches.

The GABH is an algorithm that is especially useful for bottleneck batching tool groups, in which case a large number of lots wait to be processed in front of the tool group.

The algorithm has much more room for improvement in terms of total weighted tardiness if the number of batches to be assigned to the parallel machines increases. Therefore, the optimal time window size for the GABH strategy is larger than the size of an optimal time window for the DBDH strategy.

6 CONCLUSIONS

In this paper, we investigated three different heuristics for batching in semiconductor manufacturing. The first heuristic is a dispatching heuristic that lacks in taking future lot arrivals into account. By contrast, the second heuristic considers future time arrivals in a certain time window. We suggest a genetic algorithm-based scheduling heuristic that uses the second heuristic to form batches, assigns the formed batches to single machines and sequences them on each single machine separately. We presented the results of computational experiments.

The proposed heuristics are considered for an integration in a more general framework of distributed (i.e., agent-based) hierarchical production control of semiconductor wafer fabs.

More research effort is also needed to treat all the necessary parameter settings in a situation dependent manner. In this research, we only considered the restricted case of two different process flows. However, our approach seems to be worth extending to the multi-product case.

REFERENCES

Akçali, E., Uzsoy, R, Hiscock, D. G., Moser, A. L., and Teyner, T. J. 2000. Alternative Loading and Dispatching Policies for Furnace Operations in Semiconductor Manufacturing: A Comparison by Simulation. In *Proceedings of the 2000 Winter Simulation Conference,* ed .J. A. Joines, R.R. Barton, K. Kang, and P. A. Fishwick, 1428-1435.

Atherton, L. F. and R. W. Atherton. 1995. *Wafer Fabrication: Factory Performance and Analysis*. Kluwer Academic Publishers, Boston, Dordrecht, London.

Cigolini, R., M. Perona, A. Portioli, and T. Zambeli. 2002. A New Dynamic Look-Ahead Scheduling Procedure for Batching Machines. *Journal of Scheduling*, 5: 185-204.

Chand, S., R. Traub, and R. Uzsoy. 1997. Rolling Horizon Procedures for the Single Machine Deterministic Total Completion Time Scheduling Problem with Release Dates. *Annals of Operations Research,* 70:115-125.

Fowler, J. W. and J. Robinson. 1995. *Measurment and Improvement of Manufacturing Capacities (MIMAC): Final Report.* Technical Report 95062861A-TR, SEMATECH, Austin, TX.

Fowler, J. W., D. T. Phillips, and G. L. Hogg. 1992. Real-time Control of Multiproduct Bulk-Service Semiconductor Manufacturing Processes. *IEEE Transactions on Semiconductor Manufacturing,* 5 (2): 158-163.

Fowler, J. W., G. L. Hogg, and D. T. Phillips. 2000. Control of Multiproduct Bulk Server Diffusion/Oxidation Processes, Part Two: Multiple Servers. *IIE Transactions on Scheduling and Logistics,* 32 (2): 167-176.

Glassey, C. R. and W. W. Weng. 1991. Dynamic Batching Heuristics for Simultaneous Processing. *IEEE Transactions on Semiconductor Manufacturing,* 4 (2): 77-82.

Habenicht, I. and L. Mönch. 2002. A Finite-Capacity Beam-Search Algorithm for Production Scheduling in Semiconductor Manufacturing. In *Proceedings of the 2002 Winter Simulation Conference,* ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 1406-1413.

Horiguchi, K., N. Raghavani, R. Uzsoy, and S. Venkatheswaran. 2001. Finite-Capacity Production Planning Algorithms for a Semiconductor Wafer Fabrication Facility. *International Journal of Production Research,* 39 (5): 825-842.

Mason, S., J. W. Fowler, and W. M. Carlyle. 2002. A Modified Shifting Bottleneck Heuristic for Minimizing Total Weighted Tardiness in Complex Job Shops, *Journal of Scheduling*, 5: 247-262.

Mönch, L., O. Rose, and R. Sturm. 2002. Framework for Performance Assessment of Shop-Floor Control Systems. In *Proceedings of the 2002 Modeling and Analysis of Semiconductor Manufacturing Conference (MASM 2002)*, ed. J. W. Fowler, J. K. Cochran, 95-100.

Mönch, L., H. Balasubramanian, J. W. Fowler, and M. E. Pfund. 2002. Minimizing Total Weighted Tardiness on Parallel Batch Processing Machines Using Genetic Algorithms. In *Proceedings of the International Symposium on Operations Research. Klagenfurt, Austria,* 205-211.

Robinson, J.K., J. W. Fowler, and J. F. Bard. 1995. The Use of Upstream and Downstream Information in Scheduling Semiconductor Batch Operations. *International Journal of Production Research*, 33 (7): 1849-1869.

Schömig, A. and J. W. Fowler. 2000. Modelling Semiconductor Manufacturing Operations. In *Proceedings of the 9th ASIM Dedicated Conference Simulation in Production and Logistics*, ed. K. Mertins and M. Rabe, 55-64.

Uzsoy, R., C.-Y. Lee, and L. A. Martin-Vega. 1992. A Review of Production Planning and Scheduling Models in the Semiconductor Industry, Part I: System Characteristics, Performance Evaluation and Production Planning. *IIE Transactions on Scheduling and Logistics*, 24: 47-61.

Vepsalainen, A. and T. E. Morton. 1987. Priority Rules and Lead Time Estimate for Job Shop Scheduling with Weighted Tardiness Costs. *Management Science*, 33: 1036-1047.

Wall, M. 1999. Galib: A C++ library of Genetic Algorithms Components. Website: http://lancet.mit.edu/ga/.

AUTHOR BIOGRAPHIES

LARS MÖNCH is an Assistant Professor in the Department of Information Systems at the Technical University of Ilmenau, Germany. He received a master's degree in applied mathematics and a Ph.D. in the same subject from the University of Göttingen, Germany. After receiving his Ph.D. he worked for two years for Softlab GmbH in Munich in the area of software development. His current research interests are in simulation-based production control of semiconductor wafer fabrication facilities, applied optimization and artificial intelligence applications in manufacturing. He is a member of GI (German Chapter of the ACM), GOR (German Operations Research Society), SCS and INFORMS. His email address is <Lars.Moench@tu-ilmenau.de>.

ILKA HABENICHT is a Ph.D. student in the Department of Information Systems at the Technical University of Ilmenau, Germany. She received a master's degree in business related engineering from the Technical University of Ilmenau, Germany. Her research interests are in production control of semiconductor wafer fabrication facilities and simulation. Her email address is <Ilka.Habenicht@tu-ilmenau.de>.