Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

A FINITE-CAPACITY BEAM-SEARCH-ALGORITHM FOR PRODUCTION SCHEDULING IN SEMICONDUCTOR MANUFACTURING

Ilka Habenicht
Lars Mönch

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

ABSTRACT

In this paper we describe a finite-capacity algorithm that can be used for production scheduling in a semiconductor wafer fabrication facility (wafer fab). The algorithm is a beam-search-type algorithm. We describe the basic features of the algorithm. The implementation of the algorithm is based on the ILOG-Solver libraries. We describe the simulation environment, which is used to evaluate the performance of the proposed algorithm. We show some results from computational experiments with the algorithm and the simulation test-bed described.

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 reentrant flows are typical for this type of production.

The production process of circuits requires that the manufacturing area is extremely clean, 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 on a single wafer on average up to 800 circuits. The circuits are made up of layers. Because of the layered nature of the circuits it is possible to group the reentrant parts of the production flows into the following groups: cleaning, metal deposition, chemical vapor deposition (CVD), photolithography, etching, 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 a common production scheduling problem in wafer fabs. Given a set of pending lots we are interested in determining start and end times for individual operations of the lots. Here, an operation is defined as a set of consecutive process steps.

In contrast to current practice in semiconductor manufacturing the suggested algorithm takes the capacity on the work centers of the wafer fab into account. We use a simulation model of a full wafer fab in order to determine the performance of our algorithm.

The paper is organized as follows. In the next section, we describe the problem under consideration in detail and we also summarize some related research. The main part of the paper deals with the description of the finite-capacity beam-search-algorithm. The suggested algorithm is basically a beam-search-algorithm as described in Fox (1994) and Fargher and Smith (1994). However, we modified the existing algorithms in a way suitable for the solution of the problem under consideration.

We describe the implementation of the algorithm using the ILOG-solver libraries. A description of the simulation test-bed used in the experiments, which is based on previous work of Mönch et al. (2002), is presented in Section 4. We present the results of computational experiments in Section 5.

2 PROBLEM DESCRIPTION AND MODEL

2.1 Lot Planning Problem

In this section, we will formulate the lot planning problem addressed in this paper. An aggregation procedure for

process steps takes place first. Operations are considered instead of single process steps. Here, an operation is defined as a set of consecutive process steps of one product. More formally, if a product P_k has a process flow $\bar{s}_k := (s_{k1},...,s_{kni})$, then we replace $\bar{s}_k$ by the sequence $\bar{o}_k := (o_{k1},...,o_{kml})$ of operations. Usually, an operation consists of three or four consecutive process steps. Note that the decision of which process steps form a single operation is influenced by the decomposition of the shop floor into different work areas.

The three main reasons for considering operations are: firstly, the problem size decreases, secondly, due to uncertainties in the real factory process it is not reasonable to work on the level of detail of single process steps, thirdly, the personal on the shop-floor is primarily interested in due date prescription (internal due dates) for lots with respect to the work area.

The problem can be formulated as follows. Given a set of lots $\{L_1,...,L_k\}$. We assume, that each lot L_i, $i = 1,...,k$, has to perform the operations $\bar{o}_i := \left(o_{in_i},...,o_{im_i}\right)$. By o_{in_i} we denote the first operation of lot L_i that has to be performed. We denote the due date of lot L_i by d_i. We are interested in determining the start date and the end date of each operation of the lot. The calculated start and end dates should take into account capacity restrictions of the shop-floor.

The calculated start and end dates are required in order to obtain due date estimates, to plan operator and tool requirements and to look ahead for doing optimal batch and setup decisions (cf. Fowler et al. 2000). The calculated start and end dates are also used to measure the performance of a work area. The end dates of single operations are used for the implementation of certain due date performance oriented dispatching rules at the shop floor. Furthermore, the start and end dates of the operations can be used to calculate the schedules for the individual work areas independently (cf. Mönch 2001).

2.2 Infinite Capacity Approach for Lot Planning

A method to determine the required start and end dates (without considering the violation of capacity constraints) is the following one (cf. Atherton and Atherton 1995). The operations of each lot are scheduled independent of the other lots. The basic problem is the determination of the amount of waiting time for the single operations, e.g., the time that the lots are waiting for processing on machines of the work area of the corresponding operation.

We assume, that lot L_i has the actual operation $o_{i\,ni}$ at time t. The planned due date $d_{out,i}$ of lot L_i can be calculated as follows

$$d_{out,i} := t + f \sum_{k=n_i}^{m_i} p_{ki} \quad , \tag{1}$$

where

f prescribed flow factor,
m_i index of the last operation operations for lot L_i,
p_{ki} pure processing time for operation o_{ik}.

The flow factor f is defined as the ratio of the (average) real process time (including time for waiting on processing) and the (average) pure processing time (i.e., the time without any waiting time) (cf. Atherton and Atherton 1995). The flow factor depends of the capacity of the system wafer fab and the current work in process (WIP) of the wafer fab.

The expected start dates (and of course end dates) for single operations can be determined in a recursive manner as follows

$$d_{k+1,i} := d_{k,i} + f\, p_{k+1i}\,. \tag{2}$$

Here, we denote by $d_{k,i}$ the end date of the k-th operation of lot L_i.

In cases where we cannot assume $d_i \neq d_{out,i}$ we have to recalculate the amount of accumulated waiting time. We use the following formula to determine the new amount of waiting time represented by the quantity h:

$$h := \frac{d_i - t}{\sum_{s=n_i}^{m_i} p_{si}}, \tag{3}$$

if $d_i - t \geq 0$ holds, otherwise a new due date has to be set. Here, we denote by t the actual time. The expected start dates (and of course end dates) for single operations have to be determined now by the following formula

$$d_{k+1,i} := d_{k,i} + h\, p_{k+1i}\,. \tag{4}$$

Note that the new due date setting and the recalculation procedure for the quantity h has to be done on a shift to shift or day to day basis.

2.3 Literature Review

We found a number of references dealing with lot planning issues of semiconductor wafer fabrication facilities in the literature. The approaches can be divided roughly in three different classes.

Operations research related (or mathematical) optimization approaches (sometimes in combination with discrete-event simulation) belong to the first class. The work of Hung and Leachman (cf. Hung and Leachman 1996) is an example of such an approach. The solution complexity of the underlying linear programs depends on the granularity of the used time buckets. However, in our case, we need at least for the shifts at the beginning of the planning horizon time buckets of fine granularity.

Liao et al. chose a Lagrange relaxation approach for lot scheduling in a wafer fab (Liao et al. 1996). This work is interesting, because, instead of working on the single process step level, the authors use an aggregation of process steps that is similar to our operation forming. However, we believe that due to the high computational burden of the Lagrange relaxation this approach is only feasible for a short planning horizon (one shift until one day).

Approaches of the second class employ MRP II type solution techniques, i.e. forwards and backwards calculation techniques for the lots. We refer, for example, to the work by Horiguchi et al. (2001) for such a method, applied to a reentrant flow manufacturing system. Methods of this type are also used in the system ReDS (Hadavi and Voigt 1994). Forward and backward planning algorithms are also included and implemented in commercial software packages used in semiconductor manufacturing.

Search heuristics from artificial intelligence in combination with the MRP II type methods form the third class. Here, we refer to the work by Fargher and Smith (1994) and by Fargher et al. (1994). They used a beam-search-algorithm in combination with backtracking steps for lot release and for the determination of schedules in an aggregated sense. The artificial intelligence techniques were originally developed in order to solve constraint satisfaction problems. Therefore, we cannot expect good optimization features.

A more recent scheduling approach with focus on rescheduling abilities is described in the work by Toba (1999).

3 BEAM-SEARCH-ALGORITHM

3.1 Finite-Capacity Beam-Search-Algorithm

The essence of a beam-search-algorithm is a suitable search tree as shown in Figure 1. Every node of this tree represents a time bucket (time period) T_m . The search tree is partitioned into levels. Nodes, i.e., time buckets, at level k have distance k-1 to the root. The beam width β_k gives the maximal number of nodes at level k . We have a certain freedom in choosing the beam width.

Suppose that an operation o_{ij} was assigned at level $(k-1)$ of the search tree to the time bucket T_m .

- Then we search for a possible time bucket for scheduling the following operation o_{ij+1} . Therefore, we branch into nodes on the level k with possible time buckets for scheduling o_{ij+1} . We try to schedule o_{ij+1} without violating the capacity restrictions given by the tool groups required for processing o_{ij+1} .
- A backtracking step has to be take place if a scheduling step in a time period at level k is not possible due to the violation of capacity constraints. That means, that the operation o_{ij} has to be removed from T_m and to be scheduled to one of the following period of T_m . The backtracking scheme allows more than one consecutive backtracking step.

The backtracking step is basic requirement in order to plan operations as far as possible into the future.

3.2 Capacity Modeling

Because of the existence of parallel tools, i.e., tools of the same functionality, the routing of the lots in the process flows takes place on the level of tool groups. Therefore, we have to work with tool groups instead of individual tools. The capacity of a machine group is the sum of the capacity of all machines of the group. The capacity of a tool group is reduced to a certain degree by taking into account times required for preventative maintenance and for machine failure.

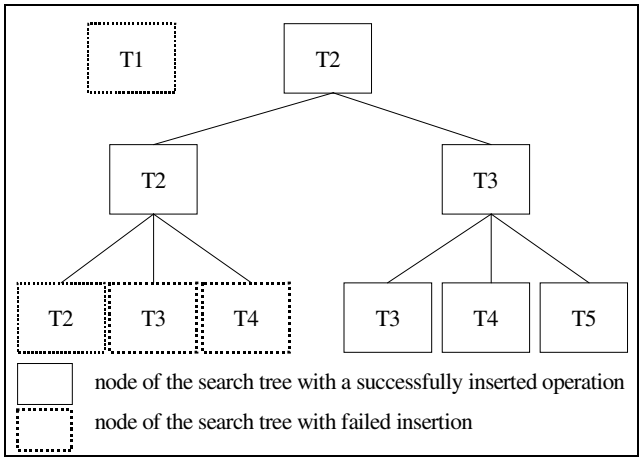

Figure 1: Search Guided by a Beam-Search-Algorithm

We are interested in scheduling operations as explained in Section 2.1. A lot is processed on different tool groups while performing an operation. In this research, we assume that for each operation it is always possible to find a tool group that behaves like a bottleneck with respect to the process steps of the operation, i.e., it is possible to construct a one-to-one mapping between an operation and a single tool group with low available capacity.

The lead time of an operation is the sum of the processing time of the process steps multiplied by a factor $\tilde{f} > 1$. This quantity $\tilde{f}$ reflects possible waiting times between the consecutive process steps of the operation. A careful adjustment of $\tilde{f}$ for different operations and product flows of the lots by using historical data from the manufacturing execution system and from simulation experiments is important for the algorithm.

3.3 Overall Structure of the Algorithm

The algorithm's goal is to determine the start and end dates for the operations of the lots. The algorithm ranks the lots by their importance (according to a certain criterion) (outer loop) and uses the beam-search-algorithm to calculate the start and end dates for the operations of the lot (inner loop). The outer loop of the algorithm is shown in Figure 2.

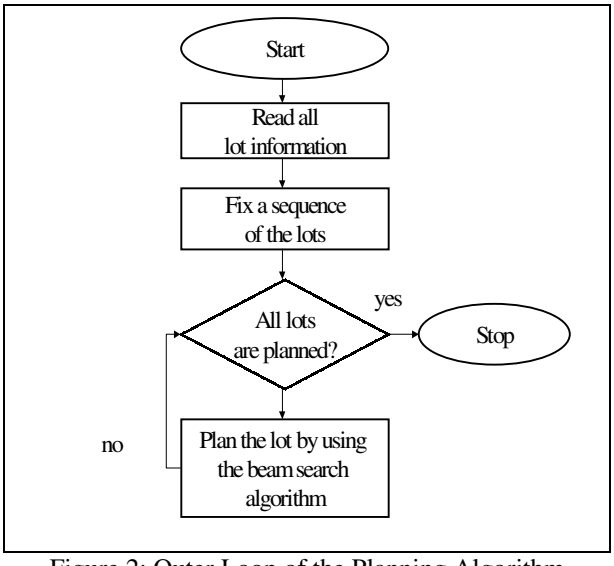

Figure 2: Outer Loop of the Planning Algorithm

The importance of a lot is determined as follows. We calculate for each lot L_i the index

$$\text{slack}_i := d_i - t - \sum_{k=n_i}^{m_i} p_{ki} , \qquad (5)$$

where we denote by t the point of time, where the calculation is performed. Then we rank the lots starting with the smallest index given by (5). W consider the due dates as an additional criterion if the slack calculated by (5) is equal for two lots.

The algorithm requires the following data input:

1. Tool and routing information, including a partition of each route into consecutive operations,
2. Forecast information,
3. Information on preventative maintenance and machine breakdowns in order to setup the available capacity of the tool groups,
4. Information on bottlenecks in order to determine the one-to-one mapping between operations and tool groups,
5. Information on the lots currently on the shop-floor and the corresponding WIP distribution.

The inner loop of the algorithm (Figure 3) works as follows. The capacity model already described in Section 3.2 is used for setting up the capacity constraints. These constraints must be formulated separately for every time bucket, for which an operation is to be scheduled. If the constraint is not satisfied in a certain time bucket, the consecutive node of the search tree will be checked. This node represents the successor of the time bucket, where the check has failed.

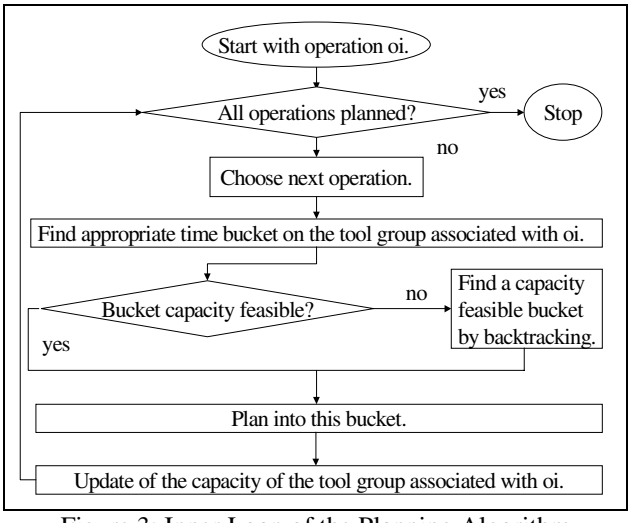

Figure 3: Inner Loop of the Planning Algorithm

At the moment, we build a new plan from scratch during each rescheduling activity. It should be possible to avoid the calculation of an entire new plan by scheduling only lots that show a deviation from the planned start dates of the current operation. However, more research is needed to work out the details.

3.4 Implementation Issues of the Algorithm

After running into problems with homemade tree libraries, mainly due to the size of the search trees, we decided to use classes from the ILOG library (Le Pape 1994).

The classes of ILOG Solver provide a framework for modeling and solving constraint programming problems and make very efficient use of memory. The library supports especially the implementation of backtracking algorithms on tree structures. For the implementation of our algorithm, we modified ILOG's implementation class IlcConstraintI and the handle class IlcConstraint, and formulated different constraints.

This way we implemented the technological constraints for the sequence of the operations and the non preemptions of the operations. The third constraint that we modeled with the help of these classes is the beam width.

Different required input data like processing times of operations, capacities of tool groups and lengths of time buckets are modeled with objects of the class IloNumArray.

4 SIMULATION MODEL

We use a discrete-event simulation tool and a simulation model of a wafer fab to evaluate the performance of our algorithm. This approach is widely used in the research community to estimate the performance of new algorithms (cf. Horiguchi et al. 2001 and Toba 1999).

We used the basic architecture described by Mönch et al. (2002) including the simulation tool AutoSched AP 7.1. The center point of this architecture is a data storage, called data model, that contains all information required in order to run the production scheduling algorithm. We extended the data model by additional classes and attributes to make it usable for the beam-search-algorithm. The basic architecture can be seen in Figure 4. Most of the data listed as input requirements of the algorithm can be read from the simulation model (as static and dynamic data).

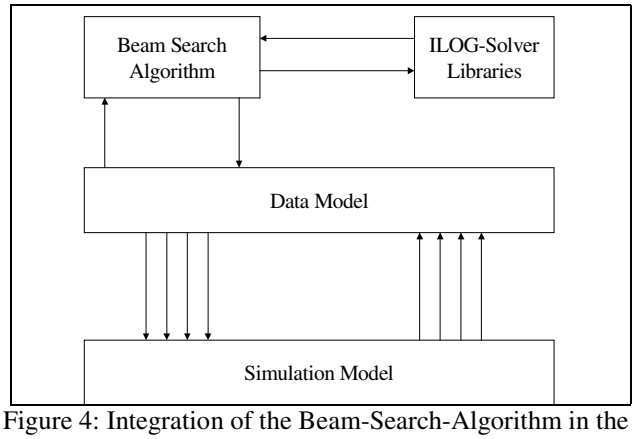

Figure 4: Integration of the Beam-Search-Algorithm in the Simulation Environment

Because the simulation tool included the AP Framework, written in the C++ programming language, the integration of the ILOG solver libraries requires only little effort.

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

We used a slack-based dispatching rule. The rule selects the lot with the smallest slack for that process step. For the calculation of the slacks of the lots waiting in front of a certain machine we use the information given by the lot plan. For that purpose we determine first the amount of time for the process steps that are corresponding to non critical machines by simply multiplying the processing time by a suitable factor. The difference between the time assigned to the operation by the lot plan and this amount of time is then used to prescribe an amount of time to the critical process step of the operation. By using this scheme it is possible to determine end dates for the single process steps.

In our experiments, we used a moderate workload of the system. Machine failures are exponentially distributed. We used a forecast horizon of three month. 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 COMPUTATIONAL RESULTS

In this section we present the results of computational experiments. We performed three different types of experiments. The following performance measures were used:

- Average tardiness of the lots released and finished within the planning horizon under consideration. We define the tardiness of lot L_i as follows:

$$T_i := \max(0, C_i^r - d_i), \tag{6}$$

  where we denote by C_i^r the realized completion time of lot L_i. In order to derive the performance measure we sum over the corresponding T_i for all lots and divide the sum by the number of lots. We denote this quantity by T_{aver}.
- Average predictability of the plans. Because the planned completion times of the lots will change over time, we use the completion time of the first plan that contains the lot as a fixed reference. We define the positive and negative deviation of the (first) plan as follows:

$$D_i^+ := \max\left\{C_i^r - C_i^p, 0\right\}, \tag{7}$$

$$D_i^- := \max\left\{C_i^p - C_i^r, 0\right\}, \quad (8)$$

where we denote by C_i^p the expected completion time of lot L_i obtained by the first plan. We sum over the quantities (7) and (8) for all corresponding lots and divide this sum by the number of lots to derive the performance measures for predictability. The corresponding quantities are denoted by D_{aver}^+ and D_{aver}^-.

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

In a first series of experiments, we are interested in answering the question, how the frequency of planning influences the system behavior. We set the length of the replanning interval as 24 hours. We used the following time bucket sizes given in Table 1 in our experiments.

Table 1: Time Bucket Settings (Scenario 1)

Time	Size of the Time Bucket
0 – 1 day	1h
1 day – 7 day	4h
7 day – 14 day	8h
14 day - end	24h

We find the results of the beam-search-algorithm in the case of four fixed bottlenecks of the line in Table 2. We consider only the bottleneck tool groups for capacity modeling. In Table 2 we present the resulting performance measures in terms of the ratio of the value obtained by the beam-search-algorithm to the corresponding value of the infinite capacity algorithm (ICA). The corresponding values for the infinite capacity algorithm described in Section 2.2 can be found in the last row of Table 2. Clearly, the algorithm is sensitive to flow factor (FF) setting. The beam-search-algorithm outperforms the infinite capacity algorithm.

Table 2: Results for Different Flow Factors based on Beam- Search-Algorithm with Four Bottlenecks

FF	T_{aver}	D_{aver}^+	D_{aver}^-	CT(A)	CT(B)	TP
1.5	0.957	0.640	12.551	1.043	0.917	1.010
1.6	0.900	0.599	13.993	1.034	0.905	1.015
1.7	0.942	0.589	17.919	1.035	0.917	1.012
ICA	1.000	1.000	1.000	1.000	1.000	1.000

In the next experiment, we fixed the flow factor and investigated the influence of a more detailed modeling of bottleneck resources. From Table 3 we can verify that an increase in the number of modeled tools also leads to an improvement of the performance measures. In Table 3 we present the resulting performance measures in terms of the ratio of the value obtained by the beam-search-algorithm to the corresponding value of the infinite capacity algorithm (ICA).

We performed a second experiment in order to evaluate the impact of the time bucket size setting to the performance measures. In this experiment, we fixed the time bucket size as 8 hours. Again, as in the first experiment, we are interested in the values of the described performance measures. We compared the two different scenarios. Scenario 2 and scenario 3 are described in Table 4.

Table 3: Results for Different Number of Modeled Bottlenecks (BN) based on Beam-Search-Algorithm (Flow Factor = 1.5)

#BN	T_{aver}	D_{aver}^+	D_{aver}^-	CT(A)	CT(B)	TP
4	0.957	0.641	12.551	1.043	0.917	1.010
14	0.910	0.725	10.503	1.036	0.905	1.016
ICA	1.000	1.000	1.000	1.000	1.000	1.000

Table 4: Different Time Bucket Settings (Scenario 2 and Scenario 3)

Time	Size of the Time Bucket
Scenario 2	
0 day – 14 day	4h
14 day - end	24h
Scenario 3	
0 day- 14 day	8h
14 day - end	24h

In Table 5 we find the performance measures of interest. It can be seen from Table 5 that we get similar results for the three scenarios. We conclude from these results that a finer grid at the beginning of the planning horizon (scenario 1) does not lead to better results, but it increases the time required for computing.

Table 5: Results for Different Time Bucket Settings with 14 Bottlenecks (Flow Factor = 1.5)

S	T_{aver}	D_{aver}^+	D_{aver}^-	CT(A)	CT(B)	TP
1	0.910	0.725	10.503	1.036	0.905	1.016
2	0.918	0.728	12.519	1.036	0.906	1.011
3	0.945	0.752	11.604	1.033	0.923	1.009
ICA	1.000	1.000	1.000	1.000	1.000	1.000

We study the impact of flow factor setting and detailed modeling of bottleneck tool groups to the performance measure average predictability of the plans in Figure 5. We present the resulting performance measures in terms of the ratio of the value obtained by the beam-search-algorithm to the corresponding value of the infinite capacity algorithm (ICA). In the case of negative deviation we divide the relative values by ten for scaling purposes. We see that the ratio of D_{aver}^- increases with increasing flow

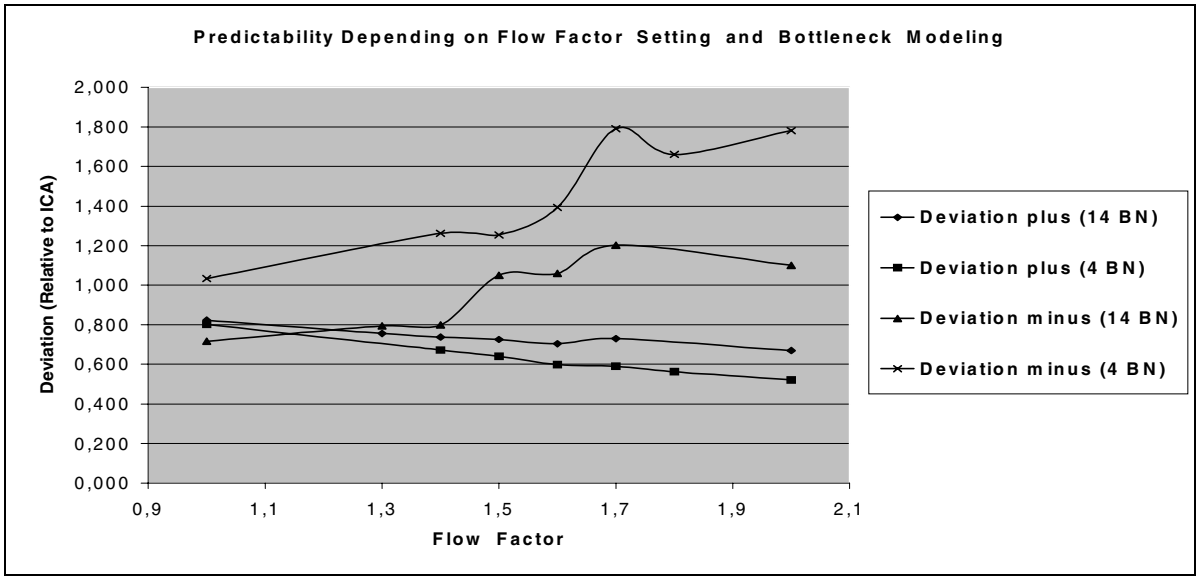

Figure 5: Impact of Flow Factor Setting and Number of Modeled Bottlenecks on the Predictability

factor. On the other hand, the ratio of D^{+}_{aver} decreases with increasing flow factor. As expected, the flow factor setting is crucial for the performance of the beam-search-algorithm. In the case of more detailed bottleneck modeling the positive deviation is not so sensitive to the flow factor setting. As expected, the proposed algorithm demonstrates some advantage compared to the infinite capacity algorithm described in Section 2.2. Our results agree with the results of Horiguchi et al. (2001). However, the authors of this paper considered a simpler algorithm that does not leave so much room for improvement.

6 CONCLUSIONS

In this paper we presented a finite-capacity beam-search for lot planning in a semiconductor wafer fab. We described the basic features of the algorithm and explained some implementation details. We presented a simulation test-bed to investigate the performance of the algorithm.

The proposed algorithm is considered for an integration in a more general framework of distributed (i.e., agent-based) hierarchical production control of semiconductor wafer fabs (cf. Vargas-Villamil and Rivera 2001 and Mönch 2001 for hierarchical and distributed production control of wafer fabs).

In the future, we will investigate different rescheduling strategies and we will consider more complex wafer fabs.

ACKNOWLEDGMENTS

The authors would like to thank Volker Schmalfuß and Bernd Friedemann, X-FAB Semiconductor Foundries AG Erfurt, Germany, for interesting discussion on the topic of this paper and for providing data sets for tests.

REFERENCES

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

Fargher, H. E., M. A. Kilgore, P. J. Kleine, and R. A. Smith. 1994. A Planner and Scheduler for Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing,* 7: 117-126.

Fargher, H. E. and R. A. Smith. 1994. Planning in a Flexible Semiconductor Manufacturing Enviroment. In *Intelligent Scheduling,* eds. M. Zweben and M. Fox, 545-581, San Francisco, CA: Morgan Kaufmann.

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., 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.

Fox, M. S. 1994. ISIS: A Retrospective. In *Intelligent Scheduling*, eds. M. Zweben and M. Fox, 3-28, San Francisco, CA: Morgan Kaufmann.

Hadavi, K. and K. Voigt. 1994. ReDS: A Real Time Production Scheduling System from Conception to Practice. In *Intelligent Scheduling,* eds. M. Zweben and M. Fox, 581-604, San Francisco, CA: Morgan Kaufmann.

Horiguchi, K., N. Raghavani, R. Uzsoy, and S. Venkatheswaran. 2001. Finite-Capacity Production Planning Algorithms for a Semiconductor Wafer Fab-

rication Facility. *International Journal of Production Research,* 39 (5): 825-842.

Hung, Y.-F. and R. C. Leachman. 1996. A Production Planning Methodology for Semiconductor Manufacturing Based on Iterative Simulation and Linear Programming Calculations. *IEEE Transactions on Semiconductor Manufacturing,* 9: 257-269.

Le Pape, C. 1994. Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems. *Intelligent Systems Engineering* 3: 55-66.

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.

Mönch, L. 2001. Towards an Agent-Based Production Control in the Semiconductor Industry. In *Proceedings 13th European Simulation Symposium– Simulation in Industry (ESS 2001),* Workshop on Multi-Agent-Based Modeling and Simulation, Marseille, 941-945.

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.

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.

Toba, H. 1999. Segment-Based Approach for Real-Time Reactive Rescheduling for Automatic Manufacturing Control. *IEEE Transactions on Semiconductor Manufacturing*, 13 (3): 264-272.

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.

Vargas-Villamil, F.-D. and D. E. Rivera. 2001. A Model Predictive Control Approach for Real-Time Optimization of Reentrant Manufacturing Lines. *Computers in Industry,* 45: 45-57.

AUTHOR BIOGRAPHIES

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. Her email address is <Ilka.Habenicht@tu-ilmenau.de>.

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) and SCS. His email address is <Lars.Moench@tu-ilmenau.de>.