PRODUCTION PLANNING IN SEMICONDUCTOR ASSEMBLY

Daniel Quadt and Heinrich Kuhn
Chair of Production, Logistics and Operations Management
Catholic University of Eichstätt-Ingolstadt
Auf der Schanz 49 - 85049 Ingolstadt
Phone: +49/841/9371823
e-mail: {daniel.quadt, heinrich.kuhn}@ku-eichstaett.de

Abstract:
The semiconductor manufacturing process usually consists of four main production stages: wafer fabrication, probe, assembly, and final test. This paper considers the lot-sizing and scheduling problem in the assembly stages. A hierarchical solution approach is outlined with focus on the first of three suggested phases, namely the bottleneck lot-sizing and scheduling phase. The approach integrates back-ordering, setup carry-over and parallel machines. A single-stage lot-sizing and scheduling problem on the bottleneck stage on product family level is solved using a new solution procedure. Core of the solution procedure is a new mixed integer programming (MIP) model that works with integer instead of binary variables. The model is embedded in a period-by-period heuristic to generate a solution to the original problem.

Keywords:
semiconductor back-end, flexible flow-line, lot-sizing and scheduling, hierarchical planning

1 Semiconductor back-end facility

The semiconductor manufacturing process usually consists of four main production stages: wafer fabrication, probe, assembly, and final test. Wafer fabrication and probe compromise the front-end, where a thin disk of silicon, the so-called wafer, is produced and tested. Each wafer contains several dies, which will later become the final product, the chip. The back-end, which combines the assembly and test operations, is often physically located at another place. In the assembly steps, the wafer is sawn and the individual chips are produced. The following test-phase inspects the individual chips before they are shipped to the costumer.

Only little research is available for back-end facilities. This is probably due to the fact that—from a technological point of view—the manufacturing process in the front-end is more complex. Nevertheless, from a logistical point of view, there are interesting and difficult problems in back-end facilities as well. Papers by Winz and Lim (2001), Sivakumar and Chong (2001) and Domaschke et al. (1998), among others, tackle back-end problems specifically.

We consider the main back-end assembly operations which are die attach, wire bonding and molding. After sewing the wafers, the die attach operation mounts the individual dies on lead frames. The wire bonding process attaches ultra-thin golden wires between each bonding pad on the die and a connector of the lead frame to create the electrical path between the die and the lead fingers. The following molding operation—often also called 'encapsulation'—encloses the individual dies in plastic or ceramic packages to protect them from the environment. After molding, subsequent production stages include Trim&Form, where the chips are severed from the lead frame and the lead fingers are formed to become the chip's legs. Finally, the separated chips are handed over to the testing operations.

Figure 1 shows a schematic view of a semiconductor back-end facility with focus on the main assembly operations.

2 Modeling the system as a flexible flow-line

Back-end assembly facilities are typically designed as flexible flow lines. A flexible flow line—also commonly referred to as hybrid flowshop, flowshop with parallel machines or multiprocessor flowshop—

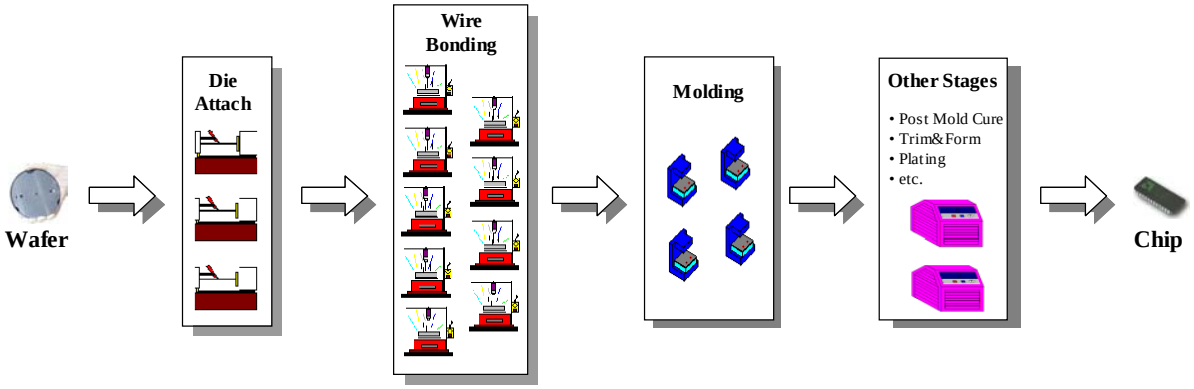

Figure 1: Schematic view of a semiconductor back-end

is a multi-level production facility with parallel machines on each stage. All products follow the same linear path through the flow line, entering the system at the first stage and leaving it after the last stage. One of the parallel machines on each stage must be selected for production. In a general form, the processing times on a stage may depend on both the machine and the product and not all machines can process all products. The number of parallel machines on each stage can vary. In between stages and before the first and after the last stage there are buffers to store the (intermediate) products.

As can be seen in figure 1, there are several die attach, wire bonding and molding machines in parallel in a semiconductor assembly facility. The products have to visit all of these three stages.

Multiple product families consisting of individual products have to be produced. For any given machine, all products of a family have the same processing time. The job processing time is significantly different between production stages. For this reason, the number of machines per stage also varies significantly—there are about eight times as many wire bonding machines as die attachers or molders. There is a deterministic, discrete demand volume for every product given in pre-defined periods.

When changing from one product to another, a setup time is incurred. Setup times are relatively long between product families (major setups, up to 12 hours) and short when changing within a product family (minor setups, ca. 1 hour). On the molding stage, no setup times incur when changing products within a product family.

For modeling purposes, we simplify the situation in the following aspects: In reality, product family setup times are sequence-dependent, but the degree of sequence-dependency is modest. It stems from the fact that product families can in turn be grouped to product family types: When changing a machine to another product family type, the family setup time is generally longer than within the product family type. As this effect is relatively small compared to the total setup time, the assumption of sequence-independent setup times is not too restrictive. Thus, we calculate average product family setup times. Product families of different types could also be combined if their processing times and setup times to other families are similar. We further assume that minor setup-times can be modeled as a prolongation of processing time. Hence, an intra-family product-to-product setup does not consume time on its own, but this time is added on an average basis to a job's processing time. Since in reality, many stochastic factors like machine breakdowns or operator availability have an impact on processing time and schedule-fulfillment, this assumption does not seems to inhibit the applicability of the approach.

Another simplification is the assumption of identical machines on the bottleneck stage: In a semiconductor back-end, the machines of a production stage can be clustered into machine-groups. Each machine-group is able to produce a specific (sub-) set of product families at a certain speed. In our approach, we assume that—on the bottleneck stage—all machines can produce all product families and the processing times depend on the product family only and not on the machine. This restriction does not prevent our approach to be used in a practical setting for two reasons: Firstly, many real-world machine-groups

are in fact physically identical machines that have just been separated by a manual planner to allow a segmentation of the problem. These pseudo machine-groups can be combined to one. Secondly, in the case of physically unidentical machines, the problem can often be clustered by solving it for each of the machine groups separately. This is possible as, regularly, the subsets of products that can be produced on different machine-groups do not overlap. On stages other than the bottleneck stage, machines need not be identical for our approach.

As the intermediate products are sufficiently small, infinite buffers between stages can be assumed. There is no transportation time between stages.

In our practical case, the number of machines on the three stages is about 25, 240 and 30, respectively. The number of considered product families is about 20 with each family combining between one and twenty active products. On average, more than 110 products have to be considered in total. Each period compromises the demand for half a week, or more than 500 product units (jobs). Production runs 24 hours a day and seven days a week.

3 Lot-sizing and scheduling problem

Hierarchical production planning approaches usually distinguish long term, medium term and short term production planning (see e.g. Hax and Candea 1984 or Drexl et al. 1994). Lot-sizing and scheduling form the short term production planning. The medium termed master production schedule delivers the input data. Its results are aggregate production quantities for end products on a product family and production facility level. Time frame is usually one to several month.

The lot-sizing phase has a more detailed view of one to several weeks. The problem is to determine in which periods to produce each product in what quantity. In the scheduling phase, the production volumes of a period—as determined by the lot-sizing phase—are assigned to individual machines. The scheduling phase also sets exact production start- and end-times for each product unit (job) and thus the job sequence on each machine.

After the scheduling phase, the final plan—consisting of exact time and machine assignments for all jobs and stages—is handed over to the shop floor.

The approach suggested here differs from traditional capacitated lot-sizing and scheduling models found in the literature in mainly three ways:

Firstly, because capacity is scarce, it might be useful to produce a product volume in a period other than its demand to save setup time. Traditional lot-sizing models allow a product to be produced in a period before its delivery to the costumer. As a consequence, inventory costs occur. In our case it is also possible that the product cannot be delivered on time. It is then back-ordered and associated back-order costs are incurred for every unit and period of the delay. Despite its importance in practical settings, only few researchers such as Millar and Yang (1994) and Cheng et al. (2001) have addressed lot-sizing problems with back-ordering.

Secondly, it is possible to carry over a setup state from one period to another. If on any machine, the last product of a period and the first of a subsequent period are the same, no setup has to be performed. If a routine leaves this unconsidered, it might not be able to find feasible solutions as too much capacity is consumed with setup time which is in fact not needed. Haase (1998) points out that solutions become significantly different when setup carry-over is considered. Setup carry-over has primarily been addressed by researchers dealing with small-bucket models involving many decision variables. Drexl and Kimms (1997) give an overview. Some authors (e.g. Dillenberger et al. 1994; Sox and Gao 1999; Gopalakrishnan et al. 2001) cover big bucket models with setup carry-over.

Thirdly, we have to consider parallel machines. Standard lot-sizing routines might suggest to conglomerate demands of subsequent periods to produce the whole volume in a single period. With only one machine and enough capacity, this method would save setup time and costs. In our case, producing more

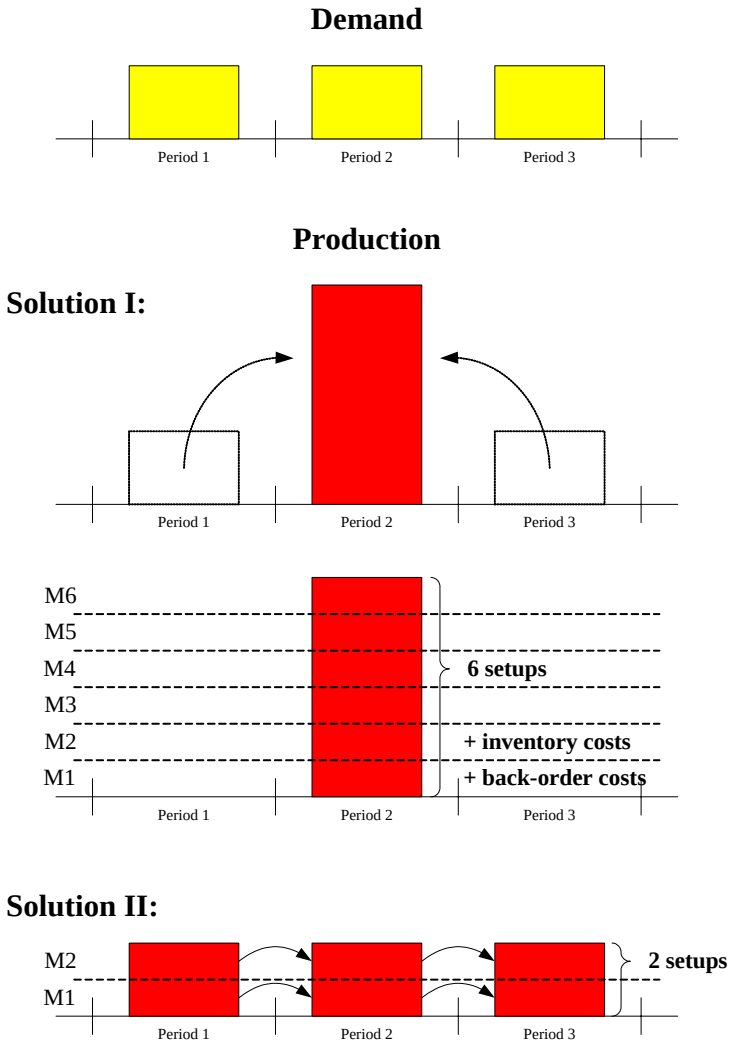

Figure 2: Parallel Machines and setup carry-over constitute a different lot-sizing problem

units in a period might lead to the result that more machines have to be set up for the product in parallel. Together with the setup carry-over, that approach would lead to higher setup times and costs. A better solution would be to produce in the demand periods, saving back-order and/or inventory costs. Fewer machines in parallel would be needed, and the setup could be carried over between the periods. Figure 2 illustrates this with an example of a single product that has an equally distributed demand over 3 consecutive periods. In solution I, 6 machines are set up for the product. Furthermore, inventory holding and back-order costs are incurred as some processing does not take place in the demand periods. Solution II produces in the demand periods, avoiding inventory holding and back-order costs. It sets up only 2 machines in parallel and their setup state is carried over from period 1 to 2 and 3. Authors including de Matta and Guignard (1995), Kang et al. (1999) and Meyr (2002) cover lot-sizing problems with parallel machines.

In the case of a (flexible) flow line with setup carry-over, lot-sizing and scheduling are intertwined: The lot-sizing phase can only determine the production volume per period when knowing the machine capacity. Machine capacity depends on the amount of time needed for setups, which in turn depends on the job sequence and setup carry-over and therefore on the scheduling phase. On the other hand, the job sequence cannot be generated without knowing which products are produced in a period, a result of the lot-sizing procedure. Ultimately, both phases have to be performed simultaneously or in a hierarchical manner.

4 Solution procedure

The lot-sizing and scheduling problem is tackled with a hierarchical planning approach described in Kuhn and Quadt (2002). It consists of the phases bottleneck planning, schedule roll-out and product-to-slot assignment.

The objective is to find a feasible schedule that minimizes all associated costs, which in our case are inventory, back-order and setup costs. Because of highly fluctuating demand and changing product specifications, the average product flow time—and hence inventory of intermediate products—shall be minimized.

The approach is bottleneck oriented. In phase 1, an integrated lot-sizing and scheduling procedure is deployed. It determines production volumes for all product families in each period and assigns machines on the bottleneck stage. The objective is to minimize setup, back-order and inventory costs of end-products. The subsequent phase rolls out the product family schedule to the other production stages. For all stages and periods considered, it determines how many machines to use for each product family and how many units to produce on each machine. It also assigns exact production times and machines to all product family units. Result are so-called machine/time-slots for each product family signalling when a product family unit is produced on a machine. The objective is to minimize the average flow-time, and thus the inventory of intermediate products. The third phase considers the individual products. It assigns products of the respective family to each of the machine/time slots from phase 2. The objective is to minimize the intra-family setups while keeping the low flow-time of phase 2.

Overall result is a detailed schedule consisting of exact time and machine assignments for each product unit in every period on all stages.

In this paper, we focus on the first planning phase, the bottleneck planning: A single-stage, multi-product capacitated parallel machine lot-sizing problem with setup carry-over and back-ordering for the bottleneck stage has to be solved. Level of detail are product families, not individual products. The problem can be modeled as a Capacitated Lot-Sizing Problem (CLSP) augmented by setup carry-over, back-orders and parallel machines. We use the following notation:

Parameters:

b_p^0	Initial back-order volume of product family p at beginning of planning interval
c_p^i	Inventory holding costs of product family p
c_p^b	Back-order costs of product family p
c_p^s	Setup costs of product family p
C	Capacity of a parallel machine per period
d_{pt}	Demand volume of product family p in period t
M	Number of parallel machines, $\bar{M} = \{1 \dots M\}$
P	Number of products families, $\bar{P} = \{1 \dots P\}$
t_p^s	Setup time of product family p
t_p^u	Per unit processing time of product family p
y_p^0	Initial inventory volume of product family p at beginning of planning interval
T	Number of periods, $\bar{T} = \{1 \dots T\}$
z	Big number, $z = \sum_{\substack{p \in \bar{P} \\ t \in \bar{T}}} d_{pt}$
ζ_{pm}^0	Initial setup state of product family p on machine m at beginning of planning interval ($\zeta_{pm}^0 = 1$, if machine m is set up for product family p)

Variables:

b_{pt}	Back-order volume of product family p at the end of period t
x_{ptm}	Production volume of product family p in period t on machine m
y_{pt}	Inventory volume of product family p at the end of period t
γ_{ptm}	Binary setup variable for product family p in period t on machine m ($\gamma_{ptm} = 1$, if a setup is performed for product family p in period t on machine m)
ζ_{ptm}	Binary linking variable for product family p in period t on machine m ($\zeta_{ptm} = 1$, if the setup state for product family p on machine m is is carried over from period t to $t+1$)

The model CLSPL-BOPM (CLSP with Linked lot-sizes, Back-Orders and Parallel Machines) can be formulated as follows:

Model CLSPL-BOPM

$$\min \sum_{\substack{p\in\bar{P}\\ t\in\bar{T}}} c_p^i y_{pt} + \sum_{\substack{p\in\bar{P}\\ t\in\bar{T}}} c_p^b b_{pt} + \sum_{\substack{p\in\bar{P}\\ t\in\bar{T}\\ m\in\bar{M}}} c_p^s \gamma_{ptm} \tag{1}$$

s. t.

$$y_{p,\,t-1} - b_{p,\,t-1} + \sum_{m\in\bar{M}} x_{ptm} - d_{pt} = y_{pt} - b_{pt} \qquad \forall\ p\in\bar{P},\ t\in\bar{T} \tag{2}$$

$$\sum_{p\in\bar{P}} \left(t_p^u x_{ptm} + t_p^s \gamma_{ptm}\right) \le C \qquad \forall\ t\in\bar{T},\ m\in\bar{M} \tag{3}$$

$$x_{ptm} \le z\left(\gamma_{ptm} + \zeta_{p,\,t-1,\,m}\right) \qquad \forall\ p\in\bar{P},\ t\in\bar{T} \tag{4}$$

$$\sum_{p\in\bar{P}} \zeta_{ptm} = 1 \qquad \forall\ t\in\bar{T},\ m\in\bar{M} \tag{5}$$

$$\zeta_{ptm} - \gamma_{ptm} - \zeta_{p,\,t-1,\,m} \le 0 \qquad \forall\ p\in\bar{P},\ t\in\bar{T},\ m\in\bar{M} \tag{6}$$

$$\zeta_{ptm} + \zeta_{p,\,t-1,\,m} - \gamma_{ptm} + \gamma_{qtm} \le 2 \qquad \forall\ p,\,q\in\bar{P},\ q\ne p,\ t\in\bar{T},\ m\in\bar{M} \tag{7}$$

$$b_{pT} = 0 \qquad \forall\ p\in\bar{P} \tag{8}$$

$$b_{p0} = b_p^0,\ y_{p0} = y_p^0,\ \zeta_{p0m} = \zeta_{pm}^0 \qquad \forall\ p\in\bar{P} \tag{9}$$

$$\begin{aligned} &b_{pt}\ge 0,\ x_{ptm}\ge 0,\ y_{pt}\ge 0,\\ &\gamma_{ptm}\in\{0,\,1\},\ \zeta_{ptm}\in\{0,\,1\} \end{aligned} \qquad \forall\ p\in\bar{P},\ t\in\bar{T},\ m\in\bar{M} \tag{10}$$

The objective function (1) minimizes the inventory, back-order and setup costs. Equations (2) are ordinary inventory flow conditions, augmented by back-order variables: The demand volume of a period t plus the back-order volume of previous periods must be met by inventory volume from previous periods or by production quantity in t. If the demand volume cannot be met, it will be back-ordered to the next period while surplus volume will be stored in inventory. Conditions (3) limit the time for processing and setups to the machine capacity. Conditions (4) ensure that a machine can only produce product family p

in a period t if the machine is set up for it. This can be achieved by either carrying over a setup state from period $t-1$ (in which case $\zeta_{p,\,t-1,\,m}=1$), or by performing a setup in period t ($\gamma_{ptm}=1$). Conditions (5) to conditions (7) handle the correct implementation of the setup carry-over. Conditions (5) imply that each machine can only carry over a setup for one product family. Conditions (6) state that only machines being set up for a certain product family can carry over a setup state for this product family to the next period. That is, if a machine m carries over a setup state for product family p from period t to $t+1$ ($\zeta_{ptm}=1$), the machine must have made a setup for this product family in period t ($\gamma_{ptm}=1$) or the setup state had been carried over from the previous period ($\zeta_{p,\,t-1,\,m}=1$). Conditions (7) deal with the following situation: If machine m carries over a setup state for product family p from period $t-1$ to t and also from t to $t+1$ ($\zeta_{p,\,t-1,\,m}=\zeta_{ptm}=1$) and in period t a setup is performed for another product family q ($\gamma_{qtm}=1$), then we have to re-set up the machine to p in period t ($\gamma_{ptm}=1$). Equations (8) ensure that the whole demand volume is produced until the end of the planning horizon, equations (9) set the initial inventory and back-order volumes and setup states. Finally, conditions (10) enforce non-negative respectively binary variables.

As this problem is intractable for practically sized instances, a heuristic is used. Core of the solution procedure is a new mixed integer programming model that works with integer variables instead of binary variables as the CLSPL-BOPM does. The new model contains only a single, aggregated resource explicitly. The integer variables are used to count the number of machines being set up for a product family. In this way, the parallel machines are handled implicitly by charging fixed-step setup costs and times occurring whenever a parallel machine has to be set up for a product family. No binary variables are needed. Since only one integer variable is needed to count all parallel machines (plus some other integer variables, e.g. to count setup carry-overs), the number of non-continuous (i.e. 'difficult') variables is reduced substantially. This holds especially when many parallel machines have to be scheduled.

Two assumptions underlie the new model: The first assumption is that a setup for a product family is only allowed if another machine producing the same product family is fully utilized and has no more capacity left over. In other words: At most one machine may be only partially loaded by a product family. This is in accordance with the behavior of a real shop-floor, as the number of setups per product family and period is kept at a low value. The second assumption is that a setup carry-over for a product family p on machine m from period t to $t+1$ is only allowed if p is the only product family produced on m in t. If more than one product family is produced on a machine in a certain period, this machine cannot carry over any setup state to the next period. In the semiconductor industry, as well as in many other real-life settings, we find high-running products utilizing complete machines and other low-running products sharing the remaining ones. We assume that the misrepresentation of reality caused by this assumption is therefore tolerable.

However, the new model cannot capture all capacity restrictions imposed by the original problem as modeled by the CLSPL-BOPM. For that reason, it is embedded in a period-by-period heuristic. In each of its iterations, a mixed-integer program (MIP) formulation of the new model is solved to optimality or near-optimality using standard algorithms (CPLEX). In the first iteration for period 1, all periods are covered by the new model. In later iterations for period t, only the periods from t to the planning horizon are covered. A scheduling sub-routine is invoked everytime an instance of the model has been solved. It tries to schedule the production volumes of the actual period as given by the solution to the model. In that way, a possible capacity misestimation is detected and—in the case of an overestimation—the surplus production volumes are postponed to the subsequent period. When there is a remaining production volume at the end of the planning horizon, a period backtracking is performed and the respective volumes are shifted towards the beginning of the planning interval.

Result of the heuristic are production volumes per period as well as machine and time assignments on the bottleneck stage—all on a product family level. These are handed over to the subsequent planning phases and a detailed schedule for all stages on individual product level is generated as described in Kuhn and Quadt (2002).

If the bottleneck stage is the last production stage, the due dates (periods of demand) within the lot-sizing and scheduling heuristic can be set according to the original delivery dates. If the bottleneck

stage is followed by other production stages, the due dates within the heuristic must be shifted forward to account for the production time on the subsequent stages. As the other stages do not constitute a bottleneck, this can be accomplished by a fixed lead time.

5 Summary and preliminary computational results

We have presented a production planning approach for a semiconductor back-end assembly facility with focus on the bottleneck lot-sizing and scheduling phase. A single-stage, multi-product capacitated parallel machine lot-sizing problem with setup carry-over and back-ordering is solved using a new solution procedure. The suggested period-by-period heuristic has a new mixed integer programming model at its core that works with integer instead of binary variables.

The stand-alone lot-sizing and scheduling algorithm has been investigated in a preliminary computational study. 1728 test instances of three different sizes have been generated systematically. We are interested in the run-time performance and the solution quality of the algorithm. The heuristic was compared to a direct implementation of the CLSPL-BOPM in CPLEX. The results are summarized in table 1. It shows the number of products, periods and machines as well as the number of instances generated for each of the three problem sets. The ratio-column indicates the average relative solution quality (costs) compared to the direct CLSPL-BOPM implementation. The time-column shows the average run-time of the heuristic.

#Products	#Periods	#Machines	#Inst	Ratio	Heuristic time	CLSPL-BOPM time
5	4	4	960	+23%	11 sec	45 sec
15	4	14	480	-7%	468 sec	600 sec (truncated)
20	6	10	288	-22%	1503 sec	3600 sec (truncated)

Table 1: Computational results for the stand-alone lot-sizing and scheduling algorithm

For the small test problems consisting of 5 products, 4 periods and 4 parallel machines, the direct CLSPL-BOPM implementation was able to find optimal solutions to most of the test instances. The heuristic leads to results that are on average about 23% worse than the optimal solutions. Bigger instances show that the heuristic outperforms the time-truncated optimal CPLEX implementation in both computation time and solution quality. The heuristic solutions for 15 products, 4 periods and 14 machines are on average 7% better than the ones generated by the direct implementation with a time limit of 10 minutes (600 seconds). For problems of 20 products, 6 periods and 10 machines, the direct implementation does not find a solution to most problems within an hour (3600 seconds) of computation time. The heuristic finds solutions in about 25 minutes on average. Compared to the solutions that could be solved with the direct implementation, the heuristic solutions are on average about 22% better.

References

Cheng, C. H., M. S. Madan, Y. Gupta, and S. So (2001). Solving the capacitated lot-sizing problem with backorder consideration. *Journal of the Operational Research Society 52*, 952–959.

de Matta, R. and M. Guignard (1995). The performance of rolling production schedules in a process industry. *IIE Transactions 27*(5), 564–573.

Dillenberger, C., L. F. Escudero, A. Wollensak, and W. Zhang (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research 75*(2), 275–286.

Domaschke, J., S. Brown, J. Robinson, and F. Leibl (1998). Effective implementation of cycle-time reduction strategies for semiconductor backend manufacturing. In D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan (Eds.), *Proceedings of the 1998 Winter Simulation Conference*, Volume 2, pp. 985–992. IEEE, Piscataway, NJ.

Drexl, A., B. Fleischmann, H.-O. Günther, H. Stadtler, and H. Tempelmeier (1994). Konzeptionelle grundlagen kapazitätsorientierter PPS-systeme. *Zeitschrift für betriebswirtschaftliche Forschung 46*(12), 1022–1045.

Drexl, A. and A. Kimms (1997). Lot sizing and scheduing - survey and extensions. *European Journal of Operational Research 99*, 221–235.

Gopalakrishnan, M., K. Ding, J.-M. Bourjolly, and S. Mohan (2001). A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management Science 47*(6), 851–863.

Haase, K. (1998). Capacitated lot-sizing with linked production quantities of adjacent periods. In A. Drexl and A. Kimms (Eds.), *Beyond manufacturing resource planning (MRP II) - advanced models and methods for production planning*, pp. 127–146. Berlin et al.: Springer.

Hax, A. C. and D. Candea (1984). *Production and Inventory Management*. Englewood Cliffs, NJ: Prentice-Hall.

Kang, S., K. Malik, and L. J. Thomas (1999). Lotsizing and scheduling on parallel machines with sequence-dependent setup costs. *Management Science 45*(2), 273–289.

Kuhn, H. and D. Quadt (2002). Lot sizing and scheduling in semiconductor assembly - a hierarchical planning approach. In G. T. Mackulak, J. W. Fowler, and A. Schömig (Eds.), *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2002)*, pp. 211–216. Arizona State University, Tempe, USA.

Meyr, H. (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research 139*(2), 277–292.

Millar, H. H. and M. Yang (1994). Lagrangean heuristics for the capacitated multi-item lot sizing problem with backordering. *International Journal of Production Economics 34*(1), 1–15.

Sivakumar, A. I. and C. S. Chong (2001). A simulation based analysis of cycle time distribution, and throughput in semiconductor backend manufacturing. *Computers in Industry 45*(1), 59–78.

Sox, C. R. and Y. Gao (1999). The capacitated lot sizing problem with setup carry-over. *IIE Transactions 31*, 173–181.

Winz, G. and P. T. Lim (2001). Assembly line planning system. In *5th Annual TAP (Test, Assembly & Packaging) Automation & Integration Conference 2001*, pp. 21–24. Semicon, Singapore.

Acknowledgement: The research is supported by grant 03KUM1EI of the Bundesministerium für Bildung und Forschung (BMBF), Germany.