Computers & Operations Research 39 (2012) 479–491

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Heuristic approaches for master planning in semiconductor manufacturing

Thomas Ponsignon [a,b], Lars Mönch [b,*]

[a] Infineon Technologies AG, Am Campeon, 85579 Neubiberg, Germany
[b] Chair of Enterprise-wide Software Systems, Department of Mathematics and Computer Science, University of Hagen, Universitaetsstrasse 1, 58097 Hagen, Germany

ARTICLE INFO

Available online 17 June 2011

Keywords:
Master planning
Semiconductor manufacturing
Decomposition
Genetic algorithm
Computational experiments

ABSTRACT

In this paper, we propose heuristic approaches for solving master planning problems that arise in semiconductor manufacturing networks. The considered problem consists of determining appropriate wafer quantities for several products, facilities, and time periods by taking demand fulfillment (i.e., confirmed orders and forecasts) and capacity constraints into account. In addition, fixed costs are used to reduce production partitioning. A mixed-integer programming (MIP) formulation is presented and the problem is shown to be NP-hard. As a consequence, two heuristic procedures are proposed: a product based decomposition scheme and a genetic algorithm. The performance of both heuristics is assessed using randomly generated test instances. It turns out that the decomposition scheme is able to produce high-quality solutions, while the genetic algorithm achieves results with reasonable quality in a short amount of time.

1. Introduction

Over the last three decades, supply chain planning topics have gained increasing popularity in industrial and academic environments (cf. [1]). Global supply chains have become a standard because of the liberalization of the world economy and progress in information technology. As a result, the scope of production planning is no longer confined to a single fabrication site, but it incorporates the concept of manufacturing networks (cf. [2]).

The purpose of production planning is to fulfill customer demands through efficient utilization of company resources. Typical production planning systems consist of four main modules organized in a hierarchical structure: (a) long-term demand planning; (b) long-term strategic allocation of production resources; (c) mid-term tactical master planning (MP), also called master production scheduling, from which capacitated production requests for final products processed within a manufacturing network are generated; (d) short-term operational planning that derives detailed schedules for jobs on the machines of a single production site (cf. [3]). The focus of this paper is on MP problems found in semiconductor manufacturing. Our main contribution consists of suggesting a suitable mathematical formulation and appropriate heuristic approaches since our problem has not previously been addressed in the literature.

The paper is organized as follows. In Section 2, we suggest an appropriate MIP formulation for our MP problem. The complexity of the problem is also analyzed. In Section 3, we discuss the related literature. Then, in Section 4 we propose two heuristic approaches: a product based decomposition scheme and a genetic algorithm. We assess the performance of both heuristics based on randomly generated test instances in Section 5. Finally, we conclude and provide directions for future research in Section 6.

2. Problem description and mathematical model

In this section, we first describe the problem of interest. Then, we introduce the necessary notation. After that, an appropriate MIP formulation is presented in the third subsection. Finally, we show that the problem investigated in this paper is NP-hard.

2.1. Problem description

Semiconductor manufacturing consists of front-end operations, where the surface of raw silicon wafers is modified to create a pattern of integrated circuits, and of back-end operations, where wafers are cut into individual microchips (dies), put in a package, and tested. In this paper, we focus on the first production stage. It takes place in wafer fabrication facilities (wafer fabs), and it requires a total processing time of up to several weeks (depending on the product complexity) to go through the several hundred required processing steps. Wafer fabrication can be performed in in-house facilities or may be outsourced to subcontractors (cf. [4–6]).

We are interested in determining appropriate wafer quantities for several products, several production sites, and several periods of time. The master plan typically has a horizon of six months divided into weekly time buckets. Since market demand is not entirely

* Corresponding author.
E-mail address: Lars.Moench@fernuni-hagen.de (L. Mönch).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.06.009

known when planning a couple of weeks or months ahead, we must distinguish between firm customer orders and additional forecasts. On the one hand, explicit customer requirements are confirmed, postponed, or reduced by the order management process based on available supply. On the other hand, the demand planning process performed every month by sales and marketing departments tries to foresee the rest of the market needs. Both are main inputs of MP (cf. [7]). In our model, orders are fulfilled at first, while forecasts are satisfied only if capacity is available due to the difference in the demand certainty. We assume that unmet customer orders in period t are carried over as backlog to period $t+1$. On the contrary, unfulfilled forecasts are ignored for the rest of the horizon. It is also assumed that inventory is possible.

Capacity modeling is a crucial point for MP. The capacity limits are related to bottleneck work centers. For each of them, we consider minimum and maximum loading bounds. The minimum threshold ensures that facilities are used even if demand is low to avoid ramp up effects (cf. [6]). We assume that all products have the same fixed cycle time. Given the completion period of a wafer and its fabrication process, we are able to compute when it arrives at a certain bottleneck work center. Next, we accumulate the time that the product spends being processing on the bottleneck machines within a specific period. Finally, the sum of machine-hours considering all wafers has to be contained within the capacity limits of the bottleneck. This method allows us to take re-entrant flows of material into account. These flows occur when wafers go through the same tools several times during their production process, which is common for front-end operations (i.e., successive steps for the modeling of integrated circuits by lithography). This capacity modeling method is similar to other approaches used for capacity planning in semiconductor manufacturing (cf. [8,9]). However, this representation is not appropriate for subcontractors since information on bottlenecks is not always available outside the company. In this case, capacity limits are converted from hours to pieces, and we simply measure the number of outsourced wafers. Note that capacity bounds are time-dependent in our problem setting. The reason is that capacity limits usually result from the monthly volume and capacity planning process, which allocates the amount of capacity to different technologies by taking tactical factors into account. As a result, the available capacity may not be constant over time. Our model also includes decisions related to the number of facilities used to fulfill the demand for one product in one period. Indeed, we consider that all items can be processed at all sites, but we are interested in minimizing production partitioning. A master plan that spreads production requests for one single product all over the manufacturing network is not economically viable due to transport costs and logistic issues. Therefore, we incorporate fixed production costs, called location costs, in our model to discourage production partitioning.

Finally, the decision problem contains an objective function that is related to the difference between revenues and total costs, and it is subject to a set of constraints. The objective function strives to keep the number of unmet customer orders low and to satisfy additional forecast demands if capacity is sufficient, whereas a small inventory level also works towards obtaining large objective function values. An inexpensive assignment of products to in-house facilities and subcontractor sites is privileged. Fixed production costs tend to keep the amount of production partitioning low.

2.2. Notation

We denote the master planning problem in semiconductor manufacturing as MPSC throughout the paper. We assume that we have a set of products $P := \{p_1,\ldots,p_{\max}\}$ that can be processed in $m_{\max}$ facilities consisting of $ih_{\max}$ in-house locations and $sc_{\max}$ subcontractor sites. The total number of bottleneck work centers associated with facilities is represented by $b_{\max}$. We assume that each bottleneck is assigned to exactly one facility and that each facility has at least one bottleneck. This assumption is reasonable because planned bottlenecks, caused by very expensive equipment, exist in all wafer fabs. We denote the number of bottlenecks in facility m by $b_{m,\max}$. Clearly $\sum_{m=1}^{m_{\max}} b_{m,\max} = b_{\max}$ holds. In case of subcontractors, we model only one bottleneck, i.e., we set $b_{m,\max} = 1$. The quantity $t_{\max}$ stands for the planning horizon measured in periods. We use one week as the length of a single time bucket. The index $k = 0,\ldots,k_{\max}$ is used to calculate the capacity consumption of the products. We assume for simplicity reasons that all products have the same cycle time of $k_{\max}+1$ weeks.

We introduce the following decision variables:

B_{pt}: backlog of confirmed orders of product p at the end of period t,
I_{pt}: inventory level of product p at the end of period t,
$s_{pt}^{(fc)}$: sales quantity of additional forecast demand of product p in period t,
$s_{pt}^{(o)}$: sales quantity of confirmed orders of product p in period t,
u_{pmt}: indicator variable for occurrence of fixed production costs of product p in facility m in period t,

$$u_{pmt} := \begin{cases} 1 & \text{if product } p \text{ is processed in facility } m \text{ in period } t, \\ 0 & \text{otherwise,} \end{cases}$$

x_{pmt}: number of wafers of product p to be completed at the end of period t in facility m.

Note that $B_{pt}, I_{pt}, s_{pt}^{(fc)}, s_{pt}^{(o)}, x_{pmt} \in \mathbb{R}$, whereas $u_{pmt} \in \{0,1\}$.
The following parameters are used in our model:

B_{p0}: initial backlog of product p at the beginning of the first period,
$C_{mbt}^{\min}$: minimum utilization of bottleneck b in facility m in period t (in hours or pieces),
$C_{mbt}^{\max}$: maximum available capacity of bottleneck b in facility m in period t (in hours or pieces),
cc_{pmbk}: capacity consumption of one wafer of product p when this product is processed in facility m at bottleneck b and the completion period is k periods ahead,
$d_{pt}^{(fc)}$: additional forecast demands for product p at the end of period t,
$d_{pt}^{(o)}$: confirmed orders for product p at the end of period t,
hc_{pt}: inventory cost for holding one wafer of product p within period t,
I_{p0}: initial inventory level of product p at the beginning of the first period,
lc_{pmt}: location cost when product p is processed in facility m in period t (i.e., fixed costs),
mc_{pmt}: cost to produce one wafer of product p in facility m in period t (i.e., variable costs),
rev_{pt}: expected revenue per wafer for satisfying additional demands of product p in period t,
udc_{pt}: cost due to unmet confirmed orders for one wafer of product p postponed from period t to period $t+1$,
$x_{pmt}^{(i)}$: initial number of wafers of product p to be completed at the end of period t in facility m, i.e., work in progress started before the first period of the model,
α: large number, i.e., $\alpha \geq \max_{m,b,t}\{C_{mbt}^{\max} / \min_{p}\{\sum_{k=0}^{\min(k_{\max}, t_{\max}-t)} cc_{pmbk}\}\}$.

2.3. Model formulation

In this subsection, we describe a MIP model for determining appropriate production quantities for several products and

several periods of time in a semiconductor manufacturing network consisting of wafer fabs as nodes. Each single wafer fab is able to process all products. In this paper, we consider a one-layer network, i.e., there is no flow of quantity across the nodes. MPSC is given by the objective function (1) and the constraints (2)–(8).

Maximize the objective function f

$$f(x,u,I,B,s^{(fc)}) := \sum_{p=1}^{p_{\max}} \sum_{t=1}^{t_{\max}} \times \left\{ rev_{pt} s_{pt}^{(fc)} - hc_{pt} I_{pt} - udc_{pt} B_{pt} - \sum_{m=1}^{m_{\max}} mc_{pmt} x_{pmt} - \sum_{m=1}^{m_{\max}} lc_{pmt} u_{pmt} \right\}, \tag{1}$$

subject to the following constraints:

$$I_{pt} = I_{p,t-1} - s_{pt}^{(o)} - s_{pt}^{(fc)} + \sum_{m=1}^{m_{\max}} x_{pmt} + \sum_{m=1}^{m_{\max}} x_{pmt}^{(i)}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{2}$$

$$s_{pt}^{(o)} + B_{pt} = d_{pt}^{(o)} + B_{p,t-1}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{3}$$

$$s_{pt}^{(fc)} \le d_{pt}^{(fc)}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{4}$$

$$C_{mbt}^{\min} \le \sum_{p=1}^{p_{\max}} \sum_{k=0}^{\min(k_{\max}, t_{\max}-t)} cc_{pmbk} (x_{p,m,t+k} + x_{p,m,t+k}^{(i)}) \le C_{mbt}^{\max}, \quad \forall m = 1,\ldots,m_{\max}, \forall b = 1,\ldots,b_{m,\max}, \forall t = 1,\ldots,t_{\max}, \tag{5}$$

$$x_{pmt} \le \alpha \cdot u_{pmt}, \quad \forall p = 1,\ldots,p_{\max}, \forall m = 1,\ldots,m_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{6}$$

$$x_{pmt} \ge 0, s_{pt}^{(o)} \ge 0, s_{pt}^{(fc)} \ge 0, I_{pt} \ge 0, B_{pt} \ge 0, \quad \forall p = 1,..,p_{\max}, \forall m = 1,..,m_{\max}, \forall t = 1,..,t_{\max}, \tag{7}$$

$$u_{pmt} \in \{0,1\}, \quad \forall p = 1,\ldots,p_{\max}, \forall m = 1,\ldots,m_{\max}, \forall t = 1,..,t_{\max}. \tag{8}$$

The objective consists in maximizing the overall difference between the revenues and the sum of costs. The first term in the objective function f models the revenues for fulfilling additional forecast demands. The costs for holding inventory are modeled by the second term. The third term refers to penalty costs for backlogged customer orders. The fourth and fifth terms represent variable and fixed production costs, respectively.

Constraint (2) represents the flow balance in every period and for every product. The inflows are the initial inventory, the production quantities, and the work in progress inventory; the outflows are the sales quantities related to confirmed orders and forecasts and the ending inventory. Constraints (3) and (4) relate sales quantities to market demand. Backlog is allowed only for customer orders. In case of additional forecasts, we only consider a maximum bound. The capacity restrictions for every bottleneck in each period are defined in constraint (5) with minimum and maximum utilization limits. The overall loading is calculated by taking production quantities and work in progress inventory of all products into account. We assume $\sum_{p=1}^{p_{\max}} cc_{pmb0} > 0$ to ensure that there is at least one product p such that $\sum_{k=0}^{\min(k_{\max}, t_{\max}-t)} cc_{pmbk} > 0$ for all $t = 1,\ldots,t_{\max}$, b, and m. Inequality (6) fixes the binary variable to 1 whenever there is a positive production for the considered product, location, and time period. On the other hand, $u_{pmt}=0$ leads to $x_{pmt}=0$. It makes sure that an additional facility is used only when it is necessary. Non-negativity and binary conditions are defined by constraints (7) and (8).

2.4. Computational complexity of MPSC

We investigate the computational complexity of MPSC. Therefore, we prove the following statement.

Proposition 1. MPSC is NP-hard.

Proof. We prove Proposition 1 by considering a special case of our original problem. First, we change MPSC to a single-product, single-facility, multi-period problem. We eliminate from f the indices related to products and facilities. We also assume revenues and inventory holding costs are equal to zero except for non-zero holding costs in the last period. Thus, we consider the following minimization formulation that is a special case of problem (1)–(8)

$$\text{Minimize} \sum_{t=1}^{t_{\max}} \{udc_t B_t + mc_t x_t + lc_t u_t + hc_t I_t\}. \tag{9}$$

In the same way, we modify the capacity restriction (5) to a single-product, single-facility, multi-period problem. We assume that the manufacturing process consists of only one bottleneck work center, i.e., $b_{1,\max} = 1$. The cycle time of the product is assumed to be one period, and thus the maximum value of time index k is zero, i.e., $k_{\max} = 0$. We obtain the capacity constraint

$$x_t \le C_t^{\max}, \quad \forall t = 1,\ldots,t_{\max}, \tag{10}$$

by setting $C_t^{\min} = 0$, $x_t^{(i)} = 0$, $\forall t = 1,\ldots,t_{\max}$, and $cc=1$. Furthermore, we choose $d_t^{(fc)} = 0$, $\forall t = 1,\ldots,t_{\max}$ and $I_0 = B_0 = 0$. We consider that the backlog costs are constant, i.e., $udc_t \equiv udc$, and higher than any other cost, i.e., $lc_t < udc$, $\forall t = 1,\ldots,t_{\max}$ and $mc_t < udc$, $\forall t = 1,\ldots,t_{\max}$. We also assume that $mc_{t_{\max}} < hc_{t_{\max}}$.

Next, we consider the following decision version of the knapsack problem (KSP) that is NP-complete (cf. [10]): Given positive integers $a_1, \ldots, a_n, A$, does there exist a subset $S \subseteq N := \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = A$ is valid? Given an arbitrary instance of KSP, we construct the following special instance of MPSC similarly to the construction of Proposition 1 in Florian et al. [11] to perform a reduction from KSP to MPSC

$$t_{\max} = n, \tag{11}$$

$$d_i^{(o)} = 0, \quad \forall i = 1,\ldots,t_{\max}-1, \tag{12}$$

$$d_{t_{\max}}^{(o)} = A, \tag{13}$$

$$lc_i = 1, \; mc_i = (a_i - 1)/a_i, C_i^{\max} = a_i, \quad \forall i = 1,\ldots,t_{\max}. \tag{14}$$

We claim that KSP has a solution if and only if there is a feasible solution of MPSC with cost at most equal to A. The production in periods $1,\ldots,t_{\max}$ has to supply the demand in period $t_{\max}$ only. Due to a demand of $d_i^{(o)} = 0$ in periods $1,\ldots,t_{\max}-1$, there is no backlog in these periods. Large backlog costs always make production preferable. Because of having non-zero holding cost only in the last period, we can always find a feasible production plan with production $\sum_{i=1}^{t_{\max}} x_i = A$ that has lower costs. In summary, we only consider solutions of the form given by (15)

$$\sum_{i=1}^{t_{\max}} x_i = A, \quad \text{with} \quad 0 \le x_i \le a_i, \quad \forall i = 1,\ldots,t_{\max}. \tag{15}$$

Because backlog can only occur in the last period, the costs of such a solution can be estimated as follows:

$$\sum_{t=1}^{t_{\max}}\{mc_t x_t + lc_t u_t + udc B_t\} = \sum_{i=1}^{t_{\max}}\left(\frac{a_i-1}{a_i}x_i + sign(x_i)\right) + udc\left(A - \sum_{i=1}^{t_{\max}} x_i\right)$$
$$\geq \sum_{i=1}^{t_{\max}} x_i + \left(A - \sum_{i=1}^{t_{\max}} x_i\right) = A. \quad (16)$$

A feasible solution with cost at most A exists if and only if $x_i \in \{0,a_i\}$, $\forall i = 1,\ldots,t_{\max}$ and $\sum_{t=1}^{t_{\max}} x_i = A$ due to the strict inequality

$$\frac{a_i-1}{a_i}x_i + 1 > x_i, \quad \forall x_i \in [0,a_i). \quad (17)$$

However, this is equivalent to the existence of a solution of KSP. Therefore, we conclude by reduction from KSP to MPSC that MPSC is NP-hard. □

As a consequence, it is necessary to look for alternative approaches since an optimum solution procedure implies a large computational burden. Obviously, the NP-hardness of the problem is due to the usage of the binary condition (8). Thus, we investigate the possibility of avoiding this constraint by defining u_{pmt} as a semi-continuous variable that has to be either zero or that takes values larger than a specified positive number (cf. [12]). However, such a formulation is not suitable for our problem as we accept only two levels of value for location costs. In addition, even if semi-continuous variables allow reducing the number of constraints in the model, the size of the branch-and-bound tree remains the same. Indeed, the MIP solver still needs to branch between the two cases where variables are either forced to be zero or constrained to be above the defined threshold. Finally, as we cannot decrease the computational burden of our problem with other modeling strategies, we have to look for efficient heuristic approaches.

3. Literature review

Production planning problems that aim to balance demand with supply in a similar way as MP problems are often addressed in the literature. We can distinguish between publications that focus on more long-term strategic capacity planning and other papers that are related to mid-term tactical or even operational production planning.

Many papers applied to semiconductor manufacturing belong to the first category. For instance, Bermon and Hood [8] suggest a linear program to optimize strategic resource allocation in semiconductor manufacturing. However, this kind of long-term planning decision does not fit with MP problems. Furthermore, in our model we have to implement alternative assignment and outsourcing decisions. This requirement is tackled in the following two papers. Stray et al. [13] and Wu and Chien [14] introduce MIP models that consider semiconductor manufacturing networks. In the first publication, an enterprise-wide resource planning approach is presented, whereas the second paper focuses on the selection of assembly outsourcing sites and order allocations. However, neither paper considers mid-term planning decisions. Moreover, Stray et al. [13] deal with aggregated product families that are not suitable for MP problems, since the optimization of master plans has to take place at a single-item level. In addition, Denton et al. [15] suggest a MIP model and corresponding heuristics for solving both strategic and operational planning decisions for a semiconductor supply chain. For this, a very detailed representation of the operations is used, which does not fit with the level of detail required for MP problems.

Publications from the second category relate MP problems to tactical decisions. For example, Chern and Hsieh [16] and Vieira and Ribas [17] introduce multi-objective MP formulations solved by heuristic approaches. However, the number of resources used to complete the production of the products is not taken into account in the objective function. Of course, it simplifies the formulation and the execution of the model since they avoid the use of binary variables, but from our point of view it is an important criterion for assignment decisions.

Zobolas et al. [18] describe a repair scheme to achieve more realistic plans when demand exceeds capacity, but their approach is not appropriate for generating entire master plans. In addition, Kallrath and Maindl [19] study the use of the SAP® APO tool for production planning in semiconductor environments. Their analysis reveals that the MP problem is solved by a purely rule-based algorithm, called Capable-to-Match (CTM). It turns out that CTM ignores the specific modeling of re-entrant flows of material. Hence, the representation of resources in CTM seems not to be appropriate for tactical and operational planning decisions in semiconductor manufacturing. In addition, no computational results are provided.

As one can see, many papers tackle similar planning problems, but none of the already proposed models explicitly addresses our specific problem. In Ponsignon et al. [20] and Ponsignon and Mönch [21], we introduce a MIP model and a genetic algorithm for MP problems in semiconductor manufacturing. But we provide only some preliminary computational results. The present paper proposes an additional product based decomposition scheme, a different chromosome representation, and an improved crossover operator for the genetic algorithm. We also present the results of additional computational experiments including larger test instances.

4. Heuristic approaches

In this section, we introduce two heuristic approaches for solving MPSC. The first method is a product based decomposition of the MIP model presented in Section 2. It is denoted by PD-MPSC throughout the paper. The second approach is a genetic algorithm that we abbreviate as GA-MPSC. The motivation for the use of each method is briefly explained. Then, the design of each algorithm is described in detail.

4.1. Product based decomposition scheme

4.1.1. Motivation for using a decomposition scheme

Because of its NP-hardness, MPSC may not be solvable to optimality in reasonable time. However, it is well-known that decomposition approaches allow reducing the computational burden for large scale production planning and scheduling problems (cf., for example, [22]). The main idea behind it is to derive several smaller sub-problems from the initial problem definition and to solve each sub-problem successively. This leads to a series of sub-solutions that are combined to obtain an overall solution.

4.1.2. Product based decomposition scheme for solving MPSC

Since most of the parameters of MPSC are product dependent, it is possible to build a feasible solution with good quality by choosing an appropriate product sequence. Therefore, the products are first sorted according to their criticality, i.e., products

with high demand of confirmed orders and high backlog costs have high priority. We consider the product of cost and demand as criticality measure to obtain a value for comparison. Then, a bundle of products is selected from the ranked list to form a sub-problem. An appropriate size of the product subsets is selected based on computational experiments (see Subsection 5.3). The reduced MPSC problem is solved by taking the demand for those products and the actual remaining capacity into account. Afterwards, the values of the decision variables are frozen, and the maximum capacity limits are decreased according to the loading that is already planned. Finally, we increment the global objective function value. The algorithm terminates when all product subsets have been considered.

The PD-MPSC scheme can be described as follows:

Step 1 Initialize the global objective function value $Z_{global}=0$.
Step 2 Sort products according to the criteria CR_p in descending order where

$$CR_p := \sum_{t=1}^{t_{\max}} udc_{pt} d_{pt}^{(o)}, \quad \forall p = 1,\ldots,p_{\max}. \tag{18}$$

Step 3 Decompose product set P into $n \geq 2$ disjoint subsets $P_1, \ldots, P_n$ of equal size, only the last subset might have a different size, such that products with similar CR_p values are gathered in the same subset or in consecutive subsets. The subsets are defined such as

$$\bigcup_{i=1}^{n} P_i = P, \quad \forall i = 1,\ldots,n, \tag{19}$$

$$P_i \cap P_j = \emptyset, \quad \forall i = 1,\ldots,n,\ \forall j = 1,\ldots,n,\ i \neq j. \tag{20}$$

Step 4 Solve MPSC given by objective function (1) and constraints (2)–(8) for the current product subset P_i by taking the actual maximum capacity limits into account and by setting the minimum capacity bounds to zero.
Step 5 Increment the global objective value with the objective value of current sub-problem

$$Z_{global} := Z_{global} + f_{P_i}(x,u,I,B,s^{(fc)}). \tag{21}$$

Step 6 Decrease the maximum capacity limits as follows:

$$C_{mbt}^{\max} := C_{mbt}^{\max} - \sum_{p \in P_i} \sum_{k=0}^{\min(k_{\max},t_{\max}-t)} cc_{pmbk}(x_{p,m,t+k} + x_{p,m,t+k}^{(i)}),$$
$$\forall m = 1,\ldots,m_{\max}, \forall b = 1,\ldots,b_{m,\max}, \forall t = 1,\ldots,t_{\max}. \tag{22}$$

Step 7 As long as any product subset has not been considered, increment the index i of the current product subset P_i and go to Step 4, else return the global objective value Z_{global}.

We see that the minimum capacity limit is ignored in the previous algorithm. Obviously, considering a minimum utilization threshold leads to an artificial increase of production quantities for products of the first subset. As a result, the remaining capacity may not be sufficient for other subsets. That is why we suggest an *a posteriori* repair loop (Steps 8–16) where the bottleneck usage in each time period is checked and increased in case that the minimum bound is not met. Indeed, the lack of production is filled in when the gap is distributed between the products. We increase the production quantities of a large number of products rather than only one or a few arbitrarily chosen products since it reduces the risk of scrap stocks. However, we pay attention to holding costs. It is also important to add production quantities only where fixed production costs are already counted. In the following, we describe the repair scheme:

Step 8 $m=0$, $b=0$, $t=0$.
Step 9 $m:=m+1$.
Step 10 $b:=b+1$.
Step 11 $t:=t+1$.
Step 12 Calculate the usage of bottleneck b of facility m in period t as follows:

$$Load_{mbt}^{(1)} := \sum_{p=1}^{p_{\max}} \sum_{k=0}^{\min(k_{\max},t_{\max}-t)} cc_{pmbk}(x_{p,m,t+k} + x_{p,m,t+k}^{(i)}). \tag{23}$$

Step 13 As long as $Load_{mbt}^{(1)} < C_{mbt}^{\min}$, repair the violation with the following step by step approach:

- Select the next product p on a list that is sorted according to average holding costs $\overline{hc}_p = \frac{1}{t_{\max}} \sum_{s=1}^{t_{\max}} hc_{ps}$ in ascending order that satisfies both conditions $x_{pmt} > 0$ and $\sum_{k=0}^{\min(k_{\max},t_{\max}-t)} cc_{pmbk} > 0$.
- Choose the quantity

$$\tilde{x} \sim U\left[0, (C_{mbt}^{\max} - Load_{mbt}^{(1)}) \Big/ \left(p_{\max} \sum_{k=0}^{\min(k_{\max},t_{\max}-t)} cc_{pmbk}\right)\right], \tag{24}$$

  where we denote by $x \sim U[a,b]$ a realization of a random variable that is uniformly distributed over $[a,b]$ for $a,b \in \Re$ and $a < b$.
- Increase the production quantity by

$$x_{pmt} := x_{pmt} + \tilde{x}. \tag{25}$$

- Increase the inventory levels for the current and the subsequent periods accordingly

$$I_{pi} := I_{pi} + \tilde{x}, \quad \forall i = t,\ldots,t_{\max}. \tag{26}$$

- If all products have been considered and $Load_{mbt}^{(1)} < C_{mbt}^{\min}$, then select the first product of the list that satisfies $\sum_{k=0}^{\min(k_{\max},t_{\max}-t)} cc_{pmbk} > 0$ and add the quantity $\tilde{x} := (C_{mbt}^{\min} - Load_{mbt}^{(1)}) / \sum_{k=0}^{\min(k_{\max},t_{\max}-t)} cc_{pmbk}$ with respect to expressions (25) and (26). Note that there always exists at least one product with this property because of the assumption in Subsection 2.3.

Step 14 If $t < t_{\max}$, then go to Step 11.
Step 15 If $b < b_{m,\max}$, then go to Step 10.
Step 16 If $m < m_{\max}$, then set $b=0$ and go to Step 9.

The setting (24) makes sure that the maximum limit is not exceeded, and the presence of $p_{\max}$ ensures that only small quantities are added in each iteration. Unsold goods are stored for the rest of the horizon as described by expression (26). The procedure stops whenever the minimum limit is reached or exceeded. It also ensures that the minimum limit is met even when none of the products satisfies the two conditions of the first sub-step within Step 13.

4.2. Genetic algorithm

4.2.1. Motivation for genetic algorithms and basic principle

It is well-known from the literature that genetic algorithms (GA) are powerful heuristics widely used for solving large scale combinatorial optimization problems from manufacturing (cf. [23]). A series of papers introduce GAs for planning or assignment problems that are somehow similar to MPSC (cf., among others, [24]). Therefore, we decide to also use a GA in this research. However, other metaheuristics seem to be appropriate too. A GA scheme starts with

a pool of chromosomes randomly spread over the search space. Each chromosome represents a solution to the considered problem instance. A fitness value derived from the objective value of the problem is assigned to each single chromosome. The set of all chromosomes from the same iteration is called a population (cf. [25]). We use sigma truncation as scaling function because it is a method that accepts negative fitness values that might occur when the sum of costs is higher than the revenues. We choose the roulette wheel method as selector, i.e., the higher the fitness the more likely a chromosome is selected as a parent. From two selected parents, two offspring are generated according to a crossover probability that states when a crossover is performed or when the parents are duplicated. Next, a random number $z \sim U[0,1]$ is assigned to each newly created chromosome. When this number is smaller than a given mutation probability, then a mutation operation takes place. Each generation the algorithm adds the offspring chromosomes to the current population, and then removes the worst members depending on their fitness to decrease the population to its original size. A replacement percentage determines the number of generated children. An overlapping population is the consequence and the resultant GA is a steady state one. The GA terminates whenever a maximum number of generations is reached or when the diversity within the population falls below a predefined threshold (cf. [26,27]).

It is crucial to design an appropriate chromosome representation, proper initialization schemes, and suitable genetic operators. Hence, in the rest of this section we present important features of GA-MPSC in more detail.

4.2.2. Chromosome representation

Each chromosome of the population of GA-MPSC represents production quantities and sales quantities of one solution to a problem instance of MPSC. Consequently, we use a three-dimensional structure as chromosome C. The chromosome C is given by

$$C_{pmt} := x_{pmt}, \quad \forall p = 1,\ldots,p_{\max}, \forall m = 1,\ldots,m_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{27}$$

$$C_{p,m_{\max}+1,t} := s^{(fc)}_{pt}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{28}$$

$$C_{p,m_{\max}+2,t} := s^{(o)}_{pt}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}. \tag{29}$$

We show the representation used in Fig. 1.

All other decision variables are deduced from the values encoded in chromosome C. For this, we use formulas (30)–(32) derived from constraints (2), (3) and (6)

$$I_{pt} := I_{pt-1} + \sum_{m=1}^{m_{\max}} x^{(i)}_{pmt} + \sum_{m=1}^{m_{\max}} C_{pmt} - C_{p,m_{\max}+1,t} - C_{p,m_{\max}+2,t}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{30}$$

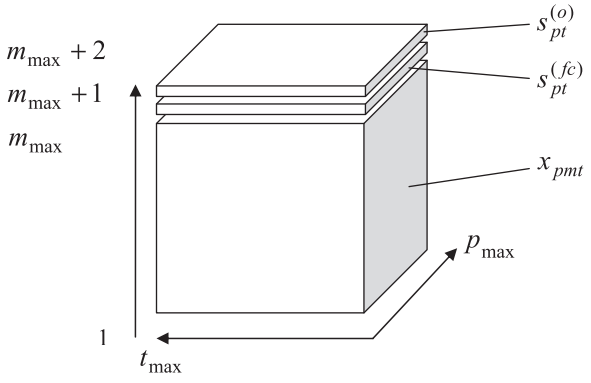

Fig. 1. Chromosome representation.

$$B_{pt} := d^{(o)}_{pt} + B_{pt-1} - C_{p,m_{\max}+2,t}, \quad \forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max} \tag{31}$$

$$u_{pmt} := sign(C_{pmt}), \quad \forall p = 1,\ldots,p_{\max}, \forall m = 1,\ldots,m_{\max}, \forall t = 1,\ldots,t_{\max}. \tag{32}$$

4.2.3. Generating the initial population

The chromosomes of the initial population are formed randomly taking the capacity restriction (5) into account. Note that it is not necessary to consider the other constraints of MPSC since they are not subject to maximum bounds, but we have to make sure that constraint (7) is fulfilled. To ensure a heterogeneous initial population we use the following three constructive schemes to determine half of the initial population:

- Scheme 1 is a simple allocation algorithm. It starts by randomly selecting a first period from $t = 1,\ldots,t_{\max}$. Orders are allocated with a higher priority than forecasts. Therefore, products are sorted in descending order in two different lists with respect to their average backlog cost $\overline{udc}_p = \frac{1}{t_{\max}} \sum_{s=1}^{t_{\max}} udc_{ps}$ and their average revenue $\overline{rev}_p = \frac{1}{t_{\max}} \sum_{s=1}^{t_{\max}} rev_{ps}$, respectively. Scheme 1 starts by considering the ranking from the first list for allocating confirmed orders, and then the second list is used for allocating additional forecast demands. Hence, the next product on the list under consideration is selected. When allocating confirmed orders, we choose the facility with the most remaining capacity, i.e., the difference between the maximum available capacity and the load already planned, among the in-house production sites as they have lower location costs. In the case that it cannot be decided between several sites because they have equal remaining available capacity, we choose the fab with the lower location cost. Moreover, if the demand for the selected product in the current time period cannot be entirely assigned to one wafer fab, an additional site, either in-house or subcontractor, with the second most remaining available capacity is selected. Afterwards, the demand is assigned to the chosen sites with respect to constraint (5). This procedure minimizes the backlog. In addition, we do not assign the production for one product in a given time period to more than two sites in order to avoid high location costs. When assigning additional forecast demands we exclusively allocate quantities to facilities where confirmed orders have already been planned. Hence, production partitioning does not generate additional costs. In the case that two facilities are used for satisfying customer orders, we first allocate the forecasts to the site with the lower variable manufacturing cost mc_{pmt} without exceeding the maximum capacity limit. If there is any remaining forecast demand it is assigned to the second site. Moreover, if capacity is not sufficient in the current period while allocating confirmed orders or additional forecast demands, the algorithm looks for available capacity in the previous periods $\max(1,t-5),\ldots,\max(1,t-1)$. This leads to pre-production and stock building, but it avoids backlog. Quantities are assigned to locations where production has already been planned so that no extra location cost is generated. If no capacity is available in the five previous periods, backlog occurs. Finally, we repeat this procedure by increasing the current period t by $t := t \bmod t_{\max} + 1$ until all products and all time periods have been considered. In addition, minimum capacity limits are taken into account. Indeed, if the loading is too low, it is increased by means of a repair loop similar to Steps 8–16 in the decomposition algorithm.
- Scheme 2 assigns random quantities to products and facilities arbitrarily chosen until the minimum utilization threshold of each bottleneck is reached in every time period.

- Scheme 3 is similar to Scheme 2 except that bottlenecks are filled until the maximum capacity limit of each bottleneck is met in every period.

The first procedure generates rather good chromosomes with respect to the objective function (1), while the second and third algorithm produce solutions with lower objective function values.

Moreover, sales quantities in chromosomes are defined with the following heuristic procedure regardless of the initialization schemes that is used:

$$C_{p,m_{\max}+2,t} = s^{(o)}_{pt} := \min\left(d^{(o)}_{pt} + B_{pt-1}, \sum_{m=1}^{m_{\max}} C_{pmt} + \sum_{m=1}^{m_{\max}} x^{(i)}_{pmt} + I_{pt-1}\right),$$

$$\forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}, \tag{33}$$

$$C_{p,m_{\max}+1,t} = s^{(fc)}_{pt} := \min\left(d^{(fc)}_{pt}, \sum_{m=1}^{m_{\max}} C_{pmt} + \sum_{m=1}^{m_{\max}} x^{(i)}_{pmt} + I_{pt-1} - C_{p,m_{\max}+2,t}\right),$$

$$\forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}. \tag{34}$$

Note that (33) and (34) both lead to $B_{pt} \geq 0$ and $I_{pt} \geq 0$, $\forall p = 1,\ldots,p_{\max}, \forall t = 1,\ldots,t_{\max}$. Based on this constructive procedure, we make sure that the initial chromosomes satisfy the capacity constraints, i.e., they are feasible solutions to a problem instance of MPSC. Note that it is important to determine the confirmed order sales quantities in a first step to ensure that only a small amount of backlog occurs. The probabilities that the first, the second, and the third schemes are used are 0.1, 0.45, and 0.45, respectively.

To achieve a diversified initial population, we use an initialization procedure based on a distance function known as heterogeneity criterion. It allows investigating a large portion of the solution space, and it enhances the optimization more than a homogenous population would. It is implemented as follows:

Step 1 Generate randomly half of the population (N is the population size).
Step 2 Initialize the index h referring to newly produced chromosomes by $h=0$.
Step 3 Generate a new chromosome C^{new} according to one of the three schemes. Again, the probabilities that the first, the second, and the third schemes are used are 0.1, 0.45, and 0.45, respectively.
Step 4 Measure the Euclidean distances between C^{new} and all the other chromosomes of the current population

$$d_{new,j}(C^{new}, C^j) := \sqrt{\sum_{p=1}^{p_{\max}} \sum_{m=1}^{m_{\max}+2} \sum_{t=1}^{t_{\max}} (C^{new}_{pmt} - C^j_{pmt})^2}, \quad \forall j = 1,\ldots,\left\lfloor \frac{N}{2} \right\rfloor + h. \tag{35}$$

Step 5 If we find at least L chromosomes from the current population for which the Euclidean distance to C^{new} is higher than $d_{\min}$ which is defined as 110% of the average Euclidean distance among chromosomes of the first half of the population:

- then add chromosome C^{new} to the population and increment $h=h+1$,
- otherwise go to Step 3, unless a predefined maximum number of attempts has been reached. Indeed, after too many unsuccessful trials, the chromosome with the highest distance is kept, h is incremented, and we proceed to Step 6.

Step 6 If $h < \lceil \frac{N}{2} \rceil$, then go to Step 3.

4.2.4. Genetic operators

4.2.4.1. Crossover. From two given parent chromosomes C^1 and C^2, we obtain two offspring chromosomes C^3 and C^4 by applying an arithmetical crossover operator. We suggest the following scheme:

Step 1 Choose two values $\lambda_1, \lambda_2 \in \Re$ as follows:

$$\lambda_1 \sim U[0,K_1],\ \lambda_2 \sim U[0,K_2] \quad \text{with} \quad K_1 \geq 1,\ K_2 \geq 1. \tag{36}$$

Step 2 $\forall p = 1,\ldots,p_{\max}, \forall m = 1,\ldots,m_{\max}+2, \forall t = 1,\ldots,t_{\max}$, set offspring as follows:

$$\begin{cases} C^3_{pmt} = \lambda \cdot C^1_{pmt} + (1-\lambda) \cdot C^2_{pmt}, \\ C^4_{pmt} = (1-\lambda) \cdot C^1_{pmt} + \lambda \cdot C^2_{pmt}, \end{cases} \quad \text{with} \quad \lambda = \begin{cases} \lambda_1, & \text{if} \quad m \leq m_{\max}, \\ \lambda_2, & \text{if} \quad m = m_{\max}+1, \\ \lambda_2, & \text{if} \quad m = m_{\max}+2. \end{cases} \tag{37}$$

It turns out that the two offspring chromosomes are created by exchanging the role of λ and $1-\lambda$. It can be easily proven that children are feasible solutions of MPSC when $\lambda_1 = \lambda_2$ and λ_1 and λ_2 are lower than or equal to 1. In fact, an arbitrary convex combination of feasible solutions of MPSC is also a feasible solution since the set determined by constraint (5) is convex. However, it is not very likely that an arithmetical crossover operator with λ values never higher than 1 can increase the objective value because parts of the objective function (1) are linear (except the first and the last term) and therefore successive generations are within the convex hull of the initial population. Hence, the offspring cannot outperform the best solution from previous generations. In addition, the last term, i.e., the location costs, can only decrease the objective value by generating more fixed production costs in the case that parent chromosomes use different facilities. In order to avoid this disadvantage by design, we propose the crossover operator presented above with λ values potentially higher than 1 in order to let the child chromosomes improve their performance and to expedite the search.

On the other hand, children obtained from (36) and (37) can be infeasible solutions of MPSC as the combination is no longer convex, i.e., capacity restriction (5) can be unfulfilled and I_{pt}, B_{pt}, and x_{pmt} can be negative. For this reason we use a penalty function as suggested by Deb [29] to make sure that any infeasible chromosome has an objective value smaller than any of the feasible chromosomes. Thereby, the probability of being selected for the next generation and for reproduction is low for infeasible solutions. Thus, to calculate the objective value of a chromosome C we use a function F that verifies if minimum and maximum capacity restrictions (5) and restriction (7) are fulfilled. We define F as follows:

$$F(C) := \begin{cases} f(x,u,I,B,s^{(fc)}), & \text{if no violation} \\ f_{\min} - \sum_{m=1}^{m_{\max}} \sum_{b=1}^{b_{m,\max}} \sum_{t=1}^{t_{\max}} \max\{0, Load^{(2)}_{mbt} - C^{\max}_{mbt}\} - \sum_{m=1}^{m_{\max}} \sum_{b=1}^{b_{m,\max}} \sum_{t=1}^{t_{\max}} \max\{0, C^{\min}_{mbt} - Load^{(2)}_{mbt}\} & \\ -\sum_{p=1}^{p_{\max}} \sum_{t=1}^{t_{\max}} \left(|\min\{0, I_{pt}\}| + |\min\{0, B_{pt}\}| + \sum_{m=1}^{m_{\max}} |\min(0, x_{pmt})| \right), & \text{otherwise} \end{cases} \tag{38}$$

with

$$Load^{(2)}_{mbt} := \sum_{p=1}^{p_{\max}} \sum_{k=0}^{\min(k_{\max}, t_{\max}-t)} cc_{pmbk}(C_{p,m,t+k} + x^{(i)}_{p,m,t+k}). \quad (39)$$

If necessary, we subtract the summation of constraint violations to the lowest objective value f_{min} of all feasible solutions in the current population. As found in our computational experiments, there are feasible solutions within each single generation. Otherwise, if there are no constraint violations, we use the objective function f as defined in Eq. (1). Finally, we derive the fitness of chromosome C from its objective function value by means of a scaling function as briefly explained in Subsection 4.2.1.

4.2.4.2. Mutation. We implement a dynamic mutation operator that allows us to continuously decrease the impact of mutation while the number of generations increases (cf. [26]). A similar technique is used by Hornung and Mönch [24]. It starts by determining a gene on chromosome C that has to be mutated. Then, in order to ensure the feasibility of the solution, an upper bound ub for the value of the gene is calculated. Note that infeasible chromosomes that result from the crossover operator are excluded from the mutation. Depending on the position of the gene to mutate, ub represents either the maximum remaining capacity or the maximum quantity that can be sold. We use the following procedure:

Step 1 Determine randomly the gene position (p^*,m^*,t^*) in chromosome C.

Step 2 Calculate ub depending on the gene position:

- If $m^* = m_{\max}+2$, then

$$ub := \min\left(d^{(o)}_{p^*t^*} + B_{p^*t^*-1}, \sum_{m=1}^{m_{\max}} C_{p^*mt^*} + \sum_{m=1}^{m_{\max}} x^{(i)}_{p^*mt^*} + I_{p^*,t^*-1}\right), \quad (40)$$

- If $m^* = m_{\max}+1$, then

$$ub := \min\left(d^{(fc)}_{p^*t^*}, \sum_{m=1}^{m_{\max}} C_{p^*mt^*} + \sum_{m=1}^{m_{\max}} x^{(i)}_{p^*mt^*} + I_{p^*,t^*-1}\right), \quad (41)$$

- If $1 \le m^* \le m_{\max}$, then

$$ub := \min_{b=1,\ldots,b_{m,\max}} \left\{ (C^{\max}_{m^*bt^*} - Load^{(2)}_{m^*bt^*}) \Big/ \sum_{k=0}^{\min(k_{\max}, t_{\max}-t^*)} cc_{p^*m^*bk} \right\}. \quad (42)$$

As one can see, Eqs. (40) and (41) are similar to expressions (33) and (34), respectively, except that the sales quantities referring to forecasts are defined independently from the sales quantities related to orders so that it increases the possible room for improvement obtained by the mutation. Indeed, it can be advantageous to decrease $s^{(o)}_{pt}$ in favor of $s^{(fc)}_{pt}$, especially if the revenues are higher than the backlog costs. Eq. (42) allows finding the bottleneck with the lowest remaining capacity. For this, we subtract the sum of load for all products as defined by formula (39) from the maximum capacity limit, and we convert the capacity into pieces by means of the capacity consumption.

In the same way, we define a lower bound lb. If the gene to be mutated refers to a sales quantity, lb is derived from constraint (7), whereas lb ensures that the minimum capacity threshold as defined in constraint (5) is respected when a production quantity has to be changed. We obtain:

Step 3 Calculate lb depending on the gene position

- If $m^* \ge m_{\max}+1$, then $lb := 0$, (43)

- If $1 \le m^* \le m_{\max}$, then

$$lb := \min_{b=1,\ldots,b_{m,\max}} \left\{ (Load^{(2)}_{m^*bt^*} - C^{\min}_{m^*bt^*}) \Big/ \sum_{k=0}^{\min(k_{\max}, t_{\max}-t^*)} cc_{p^*m^*bk} \right\}. \quad (44)$$

Based on Steps 2 and 3, we obtain an upper bound ub and a lower bound lb for the value of the gene $C_{p^*m^*t^*}$. Then, the following scheme is used to perform the dynamic mutation:

Step 4 Choose $\mu \sim U[0,1]$ and $r \sim U[0,1]$. (45)

Step 5 Determine the new value of the selected gene by

$$C^{mut} := \begin{cases} C_{p^*m^*t^*} + \Delta(n, ub - C_{p^*m^*t^*}), & \text{if } \mu \le 0.5 \\ C_{p^*m^*t^*} - \Delta(n, C_{p^*m^*t^*} - lb), & \text{otherwise} \end{cases}, \quad (46)$$

where $\Delta(n,\theta) := \theta \cdot r \cdot \left(1 - \frac{n}{\tau}\right)^b$. τ denotes a predefined maximum number of generations and n refers to current generation. The parameter b is a calibration factor.

Step 6

$$C_{p^*m^*t^*} := C^{mut}. \quad (47)$$

The function $\Delta(n,\theta)$ returns a value in the range $[0,\theta]$ such that the probability of $\Delta(n,\theta)$ being close to 0 increases as the number of completed generations n goes up. Hence, it ensures that the mutation operator searches the space uniformly at the beginning of GA-MPSC and very locally at later stages. A higher value of b leads to a lower diversifying effect of the dynamic mutation on the search space with an increasing number of generations. Since the production quantities and the sales amounts encoded in chromosome C are interlinked, changing the value of a gene also affects the other variables. Thus, a final step is required to guarantee the feasibility of the mutated solution.

Step 7 Update other related variables depending on the gene position:

- If $m^* = m_{\max}+2$, then update $C_{p^*,m_{\max}+1,t^*}$ based on equation (34).
- If $m^* = m_{\max}+1$, then update $C_{p^*,m_{\max}+2,t^*}$ as follows:

$$C_{p^*,m_{\max}+2,t^*} = s^{(o)}_{p^*t^*} := \min\left(d^{(o)}_{p^*t^*} + B_{p^*t^*-1}, \sum_{m=1}^{m_{\max}} C_{p^*mt^*} + \sum_{m=1}^{m_{\max}} x^{(i)}_{p^*mt^*} + I_{p^*t^*-1} - C_{p^*,m_{\max}+1,t^*}\right). \quad (48)$$

- If $1 \le m^* \le m_{\max}$ then update $C_{p^*,m_{\max}+1,t^*}$ and $C_{p^*,m_{\max}+2,t^*}$ based on expressions (33) and (34).

4.2.5. Improving performance by using local search

In order to enhance the intensification of the search, we implement a local search algorithm. It takes place for half of the chromosomes after generating the initial population of GA-MPSC. It also occurs regularly during the GA, i.e., every hundred generations for the ten fittest chromosomes. The frequency and extent of the local search scheme have to be appropriately chosen to find the best trade-off between computational effort and speed of convergence.

The local search routine that is implemented consists of swapping allocated production quantities to satisfy forecast demands rather than confirmed orders whenever this improves the objective function value. We assume that large revenues lead to large backlog costs. Then, we have a subset of products of size

$\lfloor \frac{p_{max}}{2} \rfloor$ with large revenues and at the same time large backlog costs, and also a second subset of products of size $\lceil \frac{p_{max}}{2} \rceil$ with low revenues and low backlog costs. We use average variable manufacturing costs $\overline{mc}_p = \frac{1}{m_{max} t_{max}} \sum_{m=1}^{m_{max}} \sum_{s=1}^{t_{max}} mc_{pms}$ as a tie breaker in case of identical revenues or backlog costs. At the beginning of GA-MPSC we establish two ranked lists such that the products of the first subset are sorted according to descending $\overline{rev}_p$ values and ascending $\overline{mc}_p$ values in the first list, while the products of the second subset are sorted according to ascending $\overline{udc}_p$ values and descending $\overline{mc}_p$ values in the second list. If two products cannot be sorted with respect to these criteria in either of the lists, their ranking order is randomly chosen. The sorted lists help us to select appropriate products for the quantity swapping. Indeed the local search procedure performs a random number of attempts defined as $\lceil \frac{p_{max}}{5} \rceil$ for each location m and each period t. At each trial, two products p_1 and p_2 are randomly selected among the first half of the first and the second list, respectively. We do not allow products to be chosen more than once for the same combination of m and t. The quantity to swap is defined in such as way that the quantities of the chromosome are not changed to a large extent. It is chosen as $x \sim U[0,0.1 \min(x_{p_2mt}, s^{(o)}_{p_2t})]$. We increase x_{p1mt} and $s^{(fc)}_{p_1t}$ by ε, and we reduce x_{p2mt} and $s^{(o)}_{p_2t}$ by ε. If the modification improves the objective function value, it is accepted, otherwise it is discarded.

Additionally, a criterion based on threshold accepting is used to avoid obtaining only a local maximum. Therefore, a modification decreasing the current objective function value can be accepted if the new objective function value is larger than a predefined threshold. We allow that the new objective function value is at most 2% smaller than the incumbent one. Within each generation of the GA, we decrease the threshold from the previous generation by 0.001%. Note that modifications violating the capacity restriction (5) are rejected. In the same manner, infeasible chromosomes resulting from the crossover operator are excluded from the local search algorithm.

We have also investigated the possibility to swap allocated quantities from earlier time periods. But due to the rather narrow difference between average backlog costs and average revenues, it is unlikely that pre-production and stock building several periods ahead will improve the objective function value. Therefore, we only swap quantities within the same time period.

5. Performance assessment of the heuristic approaches

First we explain the assessment methodology and then we describe the implementation of the different methods. Afterwards, we deal with details of the parameter settings and introduce the scheme used to generate the test instances. Finally, we present and discuss the results of computational experiments.

5.1. Assessment methodology

We assess the performance of both suggested heuristic approaches by comparing the corresponding results with the results of a commercial MIP solver after a predefined computing time (e.g., 30 min, 2 h) for different randomly generated test instances. The MIP solver is based on a branch-and-bound algorithm. It is called BB-MPSC throughout the rest of the paper when it is used for solving instances of MPSC. The comparison between the three methods is performed with respect to both solution quality and computing time.

5.2. Implementation

The BB-MPSC and the PD-MPSC procedures are performed by using the commercial MIP solver ILOG CPLEX 11.1. GA-MPSC is implemented by means of the object-oriented framework GALib 2.47 using the C++ programming language (cf. [27]). The chromosome representation suggested in Subsection 4.2.2 is coded by using the template class GA3DArrayGenome. All algorithms are tested on a computer with a 1.7 GHz Intel Pentium M processor and 1.0 GB memory.

5.3. Parameter settings

Default settings of ILOG CPLEX 11.1 are used for solving test instances with the optimum solution procedure and the decomposition scheme. Extensive computational experiments in combination with a trial and error strategy are carried out to select the best parameter combination for GA-MPSC. The parameter settings of both heuristics are summarized in Table 1.

A rather large value is chosen for the mutation probability in GA-MPSC. This is necessary to ensure that the diversity within the population does not decrease too quickly and to avoid being able to find only a local maximum. The number of products per subproblem in PD-MPSC is determined by searching the factor combinations from which the MIP solver can no longer find an optimal solution in a small amount of time, i.e., less than 2 min. It turns out that only instances with at maximum four products can be optimally solved within the given amount of time (see Table 6).

5.4. Design of experiments

We use a factorial design with two factors for the purpose of generating problem instances to compare the different algorithms. The number of products p_{max} and the number of locations m_{max}, i.e., the sum of the number of in-house sites ih_{max} and the number of subcontractors sc_{max}, are selected as factors. Each of

Table 1
Parameter settings of GA-MPSC and PD-MPSC.

Parameter settings for GA-MPSC	
Population size	$N=200$
Maximum number of generations	$\tau=1000$
Diversity threshold	0.001
Crossover probability	0.80
Mutation probability	0.07
Replacement probability	0.50
Upper bounds for arithmetical crossover	$K_1=1.25$, $K_2=1.50$
Calibration factor for dynamic mutation	$b=1.50$
Maximum attempts for the heterogeneity criterion	12
Minimum number of chromosomes with a large distance to the current population	$L=7$
Initial deterioration threshold for the local search	2%
Parameter settings for PD-MPSC	
Number of products per sub-problem	4

these factors is varied at three and four levels, respectively. Thus, we obtain 12 possible factor combinations. For each factor combination, twenty independent instances are generated and solved by the MIP solver and the heuristics. Thus, each method is applied to 240 problem instances. In addition, GA-MPSC is performed ten times for each instance with different seed values and also ten independent runs of the repair loop are carried out for PD-MPSC.

The levels of the MPSC factors are empirically determined based on the example of Infineon Technologies. However, these levels are not real-world values because of the complex data extraction and data privacy, but they reliably reflect the network structure, demand levels, available capacity, and cost configuration of a medium scale semiconductor manufacturer like Infineon. Although the costs in MPSC are defined as time-dependent we assume that they are constant over time for the computational experiments. The used design of experiments is summarized in Table 2.

5.5. Computational experiments

5.5.1. Presentation of results

The solution quality is measured as the ratio of the objective function value for both heuristic algorithms and the corresponding objective function value obtained with BB-MPSC. The measure assessing the performance of PD-MPSC is called PD/BB ratio, whereas the measure related to GA-MPSC is named GA/BB ratio. The ratio values represent the decrease or increase in percents of the objective function value by using a heuristic algorithm instead of the optimum solution procedure. All result ratios are grouped according to the factor levels of the design of experiments. The best, average, and worst ratio values as well as the confidence intervals for the average are shown in Table 3. The corresponding computing times are presented in Table 5.

The average values of the MIP gap given by the MIP solver are also shown in Table 3. It is the difference between 1 and the ratio of the objective function value of the best feasible solution found during the branch-and-bound procedure and the objective function value of the best node among the remaining solutions not necessary feasible that have not been explored within the given computing time. This percentage indicates how far the solution given by the MIP solver is from a hypothetically higher optimal solution.

5.5.2. Comparison of solution quality

Table 3 presents the grouped best, average, and worst result ratios for solving problem instances of MPSC. The average MIP gap is around 2.30% and 10.96% for problem instances with 50 and 100 products, respectively. In both cases, the product based decomposition scheme achieves results that are very close to the results obtained by BB-MPSC. Indeed the average PD/BB ratio shows an average decrease of the objective function value of around 2.23%. In addition, the results of PD-MPSC are highly homogenous since the differences between worst and best objective function values are never higher than 3.50%. The GA reaches on average 92.87% and 90.66% of the objective function values obtained by BB-MPSC for problem instances with 50 and 100 products, respectively. Note that the best average GA/BB ratio is achieved for instances with 50 products, six in-house sites, and four subcontractors by a value of 96.82%. The results of GA-MPSC

Table 2
Factors of MPSC and their different levels.

Factor	Symbol	Level for MPSC	Number of values
Number of time periods	$t_{\max}$	24	1
Number of products	$p_{\max}$	50, 100, 200	3
Number of locations	$m_{\max}$	8, 10, 10, 12	
In-house sites		6, 8	2
Subcontractor sites		2, 4	2
Number of bottlenecks per location	$b_{m,\max}$	1	1
Confirmed order	$d_{pt}^{(o)}$	$U[200,300]\cdot m_{\max}$	1
Additional forecast	$d_{pt}^{(fc)}$	$U[200,300]\cdot m_{\max}$	1
Initial inventory	I_{p0}	$500\cdot m_{\max}/p_{\max}$	1
Initial backlog	B_{p0}	$250\cdot m_{\max}/p_{\max}$	1
Work-in process			
-In-house sites	$x_{pmt}^{(i)}$	$400/p_{\max}$	1
-Subcontractor sites		$200/p_{\max}$	
Maximum available capacity			
In-house sites	$C_{mbt}^{\max}$	6720 hours	1
Subcontractor sites		2000 wafers	
Minimum capacity limit	$C_{mbt}^{\min}$	$C_{mbt}^{\min}=0.80\cdot C_{mbt}^{\max}$	1
Capacity consumption			
In-house sites	cc_{pmbk}	2 h/wafer	1
Subcontractor sites		1 wafer	
Product cycle time	$k_{\max}+1$	6 periods	1
Variable manufacturing cost		Each range is for 50% of the products	
In-house sites	mc_{pmt}	$U[10,20]$, $U[20,40]$	
Subcontractor sites		$U[30,40]$, $U[40,60]$	1
Fixed location cost		$U[375,625]$	1
In-house sites	lc_{pmt}	$U[625,1250]$	
Subcontractor sites			
Inventory holding cost	hc_{pt}	$U[5,10]$	1
Cost due to unmet demand	udc_{pt}	Each range is for 50% of the products	
		$U[200,240]$, $U[300,400]$	1
Revenue for fulfilling additional forecast demand	rev_{pt}	Each range is for 50% of the products	
		$U[80,120]$, $U[150,200]$	1
Total parameter combinations			12
Number of problem instances per combination			20
Total number of problem instances			240

Table 3
Average ratio values and confidence intervals of PD/BB and GA/BB with a level of confidence of 95%, minimum and maximum ratio values, and average MIP gaps for different factor levels as defined in Table 2.

Products p_{max}	Locations		PD/BB ratio			GA/BB ratio			Average
	ih_{max}	sc_{max}	CI	Min	Max	CI	Min	Max	MIP Gap
50	6	2	0.9775 ± 0.0028	0.9656	0.9877	0.9151 ± 0.0038	0.8816	0.9476	0.0318
50	6	4	0.9782 ± 0.0026	0.9701	0.9897	0.9682 ± 0.0025	0.9489	0.9843	0.0216
50	8	2	0.9774 ± 0.0027	0.9656	0.9883	0.9164 ± 0.0028	0.8802	0.9462	0.0200
50	8	4	0.9753 ± 0.0037	0.9590	0.9895	0.9152 ± 0.0027	0.8776	0.9518	0.0185
100	6	2	0.9824 ± 0.0041	0.9643	0.9984	0.9023 ± 0.0030	0.8823	0.9300	0.1441
100	6	4	0.9743 ± 0.0026	0.9656	0.9875	0.9044 ± 0.0044	0.8689	0.9609	0.1099
100	8	2	0.9807 ± 0.0025	0.9702	0.9884	0.9190 ± 0.0036	0.8821	0.9696	0.1056
100	8	4	0.9762 ± 0.0030	0.9640	0.9869	0.9007 ± 0.0061	0.8087	0.9374	0.0786
200	6	2	1.1343 ± 0.0156	1.0827	1.1968	1.0137 ± 0.0054	0.9728	1.0718	0.7361
200	6	4	1.1259 ± 0.0079	1.0958	1.1659	1.0100 ± 0.0088	0.9850	1.0413	0.6879
200	8	2	1.1092 ± 0.0078	1.0900	1.1487	0.9828 ± 0.0038	0.9536	1.0191	0.4781
200	8	4	1.1090 ± 0.0093	1.0612	1.1433	0.9836 ± 0.0041	0.9650	1.0146	0.4008
Overall			1.0250 ± 0.0105	1.0045	1.0476	0.9443 ± 0.0033	0.9089	0.9812	0.2361

have a mean span of 7.47% between worst and best outcomes, and the maximum difference reaches 12.87%.

The average MIP gap for instances with 200 products is 57.57%. This indicates that the MIP solver is unable to deliver high-quality solutions within the given amount of time due to the increased problem size. In fact, the decomposition scheme on average outperforms BB-MPSC for large scale problem instances with 200 products. Indeed PD-MPSC reaches a mean performance of 111.96%, whereas the mean GA/BB ratio is around 99.75%, i.e., GA-MPSC and BB-MPSC obtain on average very similar results.

An analysis of variance at the 5% level is performed to investigate the influence of the design factors, i.e., number of products, number of locations, and the different heuristics, on the solution quality relatively to the BB-MPSC values after performing a model adequacy check. Departures from normality and from constant variance are moderate. It turns out that the number of products and the different heuristics have an impact of the solution quality. Furthermore, there is an interaction between the different heuristics and the number of products.

Columns 4 and 7 in Table 3 show the 95% confidence intervals for the mean of PD/BB and GA/BB ratios, respectively. The rather narrow intervals indicate that differences between both heuristics are systematic, given the chosen level of confidence, since no overlapping occurs. The results of a Wilcoxon signed-rank test (cf. [28]) with a significance level of 1% are shown in Table 4. With respect to the objective function values achieved by the different algorithms we obtain that BB-MPSC > PD-MPSC > GA-MPSC for instances with 50 and 100 products, and PD-MPSC > BB-MPSC > GA-MPSC for instances with 200 products.

Table 4
Results of the Wilcoxon signed-rank test with 1% significance level.

Products p_{max}	Locations		PD-MPSC vs.BB-MPSC	GA-MPSC vs.BB-MPSC	PD-MPSC vs.GA-MPSC
	ih_{max}	sc_{max}			
50	6	2	<	<	>
50	6	4	<	<	>
50	8	2	<	<	>
50	8	4	<	<	>
100	6	2	<	<	>
100	6	4	<	<	>
100	8	2	<	<	>
100	8	4	<	<	>
200	6	2	>	<	>
200	6	4	>	<	>
200	8	2	>	<	>
200	8	4	>	<	>

'>' indicates that the first algorithm mentioned in the headline performs significantly better than the second mentioned algorithm; and '<' indicates the opposite.

Table 5
Average computing times (in min) of PD-MPSC, GA-MPSC and BB-MPSC for different factor levels as defined in Table 2.

Number of products p_{max}	Number of locations		Avg. computing time (in min)		
	ih_{max}	sc_{max}	PD-MPSC	GA-MPSC	BB-MPSC
50	6	2	10	3	30
50	6	4	10	3	30
50	8	2	10	4	30
50	8	4	10	4	30
100	6	2	15	6	30
100	6	4	15	6	30
100	8	2	15	7	30
100	8	4	15	7	30
200	6	2	30	9	30
200	6	4	30	10	30
200	8	2	30	10	30
200	8	4	30	10	30

5.5.3. Comparison of computing times

Besides the solution quality we assess the performance of both heuristic approaches with respect to computing time. One can argue that time is not a critical parameter for mid-term planning activities that are weekly executed; on the contrary we believe that short computing times increase the acceptance of supply chain managers and planners for those methods, and that the algorithms can be more easily implemented in planning processes.

As shown in Table 5, the predefined maximum computing times are 30 min for BB-MPSC and 10, 15 and 30 min for PD-MPSC for instances with 50, 100 and 200 products, respectively. On the contrary, GA-MPSC terminates whenever one of the termination criteria is fulfilled, as explained in Subsection 4.2.1. As one can see GA-MPSC is the fastest algorithm as it performs in 6.58 min on average. For large scale test instances, it needs only one third of the time required by the other methods. Note that the computing time of GA-MPSC increases with the number of products. Indeed, the extended chromosome structure raises the time consumption of initialization and evaluation procedures.

5.5.4. Additional experiments

In order to further assess the performance of both heuristics, we carry out additional experiments for a reduced number of small size and large size test instances. Small size instances are generated with respect to the design of experiments summarized in Table 2. Note that the maximum available capacity is reduced according to the number of products, i.e., 350 and 700 in-house hours as well as 100 and 200 maximum outsourced wafers, respectively, for instances with 1 to 5 products, and 10 products. For small and large size problems, ten and twenty independent instances are performed, respectively, for each factor combination. The average values are presented and discussed in the following.

Table 6 shows that instances with a maximum of four products can be optimally solved in a few seconds by the MIP solver. We notice an increase of hardness for instances with at least five products where PD-MSC achieves high-quality results in much less time than BB-MPSC, i.e., the average computing times of PD-MPSC are below one minute, while BB-MPSC runs for 30 min. For small size test instances GA-MPSC cannot outperform the MIP solver in terms of time, but it does when the problem size increases. Its average solution quality is also high, i.e., 99.22%.

Table 7 presents the PD/BB and GA/BB ratio values for a reduced number of large size test instances, i.e., with 200 products as described in Table 2, when the BB-MPSC runs two hours. Although the average MIP gap is still high, the extended computing time allows the MIP solver to achieve slightly better objective function values. Thus, the performance of both heuristics decreases compared to Table 3, but the average solution quality is still high, i.e., 106.33% and 95.49% for PD-MPSC and GA-MPSC, respectively.

6. Conclusions and future research

A model formulation for MP problems in semiconductor manufacturing is presented in this paper. We investigated the computational complexity of the problem. As a consequence of the NP-hardness, we presented two heuristic approaches: a product based decomposition scheme and a GA. We described in detail the problem-specific designs and operators. Many experiments were accomplished to assess the performance of both methods. We concluded that PD-MPSC provides excellent results in term of solution quality compared to the branch-and-bound procedure, whereas GA-MPSC achieves a reasonable solution quality and clearly outperforms the other methods with respect to computing time.

There are two future research directions. It is well known that production planning problems are complicated by the interaction between cycle times and resource utilization (cf. [30]). In fact, cycle times are used as inputs to determine the timing of production releases, but they are also influenced by the product mix that results from the planning activities. In this paper, we assumed fixed cycle times. Although this assumption makes sense for highly aggregated strategic planning problems, it is not desirable for tactical decisions. Hung and Leachman [31] first introduce an iterative simulation-optimization scheme, where a simulation model estimates cycle times that are used in turn as input parameters in an optimization model. Irdem et al. [32] investigate the convergence of this approach under several experimental conditions. In a complementary approach, clearing functions are discussed by Irdem [33] for the same problem. In future research, we are interested in implementing our MP model in a similar framework to obtain more realistic cycle times.

Table 6
Solution quality and average computing times (in S) of PD-MPSC and GA-MPSC, and average MIP gaps for small size test instances.

Number of products p_{max}	Number of locations		Average PD/BB ratio	Average GA/BB ratio	Average MIP gap	Avg. computing times (in s)		
	ih_{max}	sc_{max}				PD-MPSC	GA-MPSC	BB-MPSC
1	6	2	1.0000	0.9996	0.0000	5	22	5
1	8	4	1.0000	0.9991	0.0000	5	27	5
2	6	2	1.0000	0.9988	0.0000	5	39	5
2	8	4	1.0000	0.9982	0.0000	5	41	5
3	6	2	1.0000	0.9972	0.0000	6	52	6
3	8	4	1.0000	0.9979	0.0000	6	49	6
4	6	2	1.0000	0.9905	0.0000	7	61	7
4	8	4	1.0000	0.9901	0.0000	7	58	7
5	6	2	0.9992	0.9856	0.0022	32	70	1800
5	8	4	0.9993	0.9878	0.0027	32	72	1800
10	6	2	0.9951	0.9804	0.0056	54	119	1800
10	8	4	0.9946	0.9811	0.0067	54	127	1800
Overall Average			0.9990	0.9922	0.0014	18.17	61.42	603.83

Table 7
Solution quality of PD-MPSC and GA-MPSC, and average MIP gaps for large scale test instances.

Number of products p_{max}	Number of locations		PD/BB ratio			GA/BB ratio			Average MIP gap
	ih_{max}	sc_{max}	Max	Avg.	Min	Max	Avg.	Min	
200	6	2	1.1601	1.0788	1.0322	1.0577	0.9701	0.9234	0.3257
200	6	4	1.0622	1.0401	1.0045	0.9701	0.9377	0.9081	0.2889
200	8	2	1.0855	1.0687	1.0399	0.9899	0.9582	0.9256	0.2194
200	8	4	1.0898	1.0655	1.0401	0.9709	0.9534	0.9408	0.2077
Overall average			1.0994	1.0633	1.0292	0.9972	0.9549	0.9245	0.2604

Computing time of BB-MPSC: 2 h; computing times of PD-MPSC and GA-MPSC: see Table 5.

Another important stream of research refers to the analysis of demand uncertainty in MP models. Most of the related publications deal with plan execution and with determining appropriate parameter settings. For example, Venkataraman and Nathan [34], Tang and Grubbström [35], Xie et al. [36], Brandimarte [37], Huang et al. [38], and Robinson et al. [39] investigate the influence of forecasting errors on rolling master plans for different lengths of frozen period and different replanning frequencies. Barahona et al. [9] and Leung et al. [40] focus on robust planning. It seems very important to incorporate our MP model in a rolling horizon setting using discrete event simulation and to analyze its nervousness when the forecast demand is no longer deterministic.

Acknowledgments

The authors would like to thank Infineon Technologies AG for financial support of this research and Hans Ehm for fruitful discussions. They also gratefully acknowledge the valuable comments and suggestions provided by the three anonymous reviewers that helped to improve the presentation and the content of the paper.

References

[1] Stadtler H, Kilger C. Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies. 4th Ed.. Springer; 2007.
[2] Chien C-F, Dauzère-Pérès S, Ehm H, Fowler JW, Jiang Z, Krishnaswamy S, Mönch L, Uzsoy R. Modeling and analysis of semiconductor manufacturing in a shrinking world: challenges and successes. In: Proceedings of the Winter Simulation Conference; 2008. p. 2009–93.
[3] Tempelmeier H. Supply chain planning with advanced planning systems. In: Proceedings of the Aegean International Conference on Design and Analysis of Manufacturing Systems; 2001.
[4] Uzsoy R, Lee C-Y, Martin-Vega LA. A review of production planning and scheduling models in the semiconductor industry part I: system characteristics, performance evaluation and production planning. IIE Transactions 1992;24(4):47–60.
[5] Uzsoy R, Lee C-Y, Martin-Vega LA. A review of production planning and scheduling models in the semiconductor industry part II: shop-floor control. IIE Transactions 1994;26(5):44–55.
[6] Gupta JND, Ruiz R, Fowler JW, Mason SJ. Operational planning and control of semiconductor wafer fabrication. Production Planning and Control 2006; 17(7):639–47.
[7] Vieira GE. Understanding Master Production Scheduling from a Practical Perspective: Fundamentals, Heuristics, and Implementations. In: Hermann JW, editor. Handbook of Production Scheduling. Springer; 2006. p. 149–76.
[8] Bermon S, Hood SJ. Capacity optimization planning system (CAPS). Interfaces 1999;29(5):31–50.
[9] Barahona F, Bermon S, Günlük O, Hood S. Robust capacity planning in semiconductor manufacturing. Naval Research Logistics 2005;52(5):459–68.
[10] Garey MR, Johnson DS. Computers and Intractability: a Guide to the Theory of NP-Completeness. San Francisco: W. H. Freeman and Co.; 1979.
[11] Florian M, Lenstra JK, Rinnooy Kan AHG. Deterministic production planning: algorithms and complexity. Management Science 1980;26(7):669–79.
[12] Voß S, Woodruff DL. Introduction to Computational Optimization Models for Production Planning in a Supply Chain. 2nd Ed.Springer; 2006.
[13] Stray J, Fowler JW, Carlyle M, Rastogi AP. Enterprise-wide semiconductor resource planning. IEEE Transactions on Semiconductor Manufacturing 2006; 19(2):259–68.
[14] Wu J-Z, Chien C-F. Modeling strategic semiconductor assembly outsourcing decisions based on empirical settings. OR Spectrum 2008;30(3):401–30.
[15] Denton BT, Forrest J, Milne RJ. IBM solves a mixed-integer program to optimize its semiconductor supply chain. Interfaces 2006;36(5):386–99.
[16] Chern C-C, Hsieh J-S. A heuristic algorithm for master planning that satisfies multiple objectives. Computers & Operations Research 2007;34(11): 3491–513.
[17] Vieira GE, Ribas PC. Fractional factorial analysis to the configuration of simulated annealing applied to the multi-objective optimization of master production scheduling problems. International Journal of Production Research 2008;46(11):3007–26.
[18] Zobolas GI, Tarantilis CD, Ioannou G. Extending capacity planning by positive lead time and optional overtime, earliness and tardiness for effective master production scheduling. International Journal of Production Research 2008; 46(12):3359–86.
[19] Kallrath J, Maindl TI. Real Optimization with SAP® APO. Springer; 2006.
[20] Ponsignon T, Habla C, Mönch L. A model for master planning in semiconductor manufacturing. Proceedings of the Industrial Engineering Research Conference 2008:1592–7.
[21] Ponsignon T, Mönch L. A genetic algorithm to solve master planning problems in semiconductor manufacturing. Proceedings of the Industrial Engineering Research Conference 2009:2097–102.
[22] Pochet Y, Wolsey LA. Production Planning by Mixed-Integer Programming. Springer; 2006.
[23] Aytug H, Khouja M, Vergara FE. Use of genetic algorithms to solve production and operations management problems: a review. International Journal of Production Research 2003;41(17):3955–4009.
[24] Hornung A, Mönch L. Heuristic approaches for determining minimum cost delivery quantities in supply chains. European Journal of Industrial Engineering 2008;2(4):377–400.
[25] Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning. Boston, MA: Kluwer Academic Publishers; 1989.
[26] Michalewicz Z. Genetic Algorithms+Data Structures=Evolution Programs. 3rd Ed.Springer; 1996.
[27] Wall M. 'GALib—a C++ Library of Genetic Algorithm Components', 〈http://lancet.mit.edu/ga/〉; 2009.
[28] Wilcoxon F. Individual comparison by ranking methods. Biometrics Bulletin 1945;1(6):80–3.
[29] Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 2000;186(2-4):311–38.
[30] Pahl J, Voß S, Woodruff DL. Production planning with load dependent lead times: an update of research. Annals of Operations Research 2007;153(1): 297–345.
[31] Hung Y-F, Leachman RC. A production planning methodology for semiconductor manufacturing based on iterative simulation and linear programming calculations. IEEE Transactions on Semiconductor Manufacturing 1996;9(2): 257–69.
[32] Irdem DF, Kacar NB, Uzsoy R. An exploratory analysis of two iterative linear programming—simulation approaches for production planning. IEEE Transactions on Semiconductor Manufacturing 2010;23(3):442–55.
[33] Irdem DF. Evaluation of clearing functions' fitting methodology and performance for production planning models, Master Thesis, North Carolina State University, Raleigh; 2009.
[34] Venkataraman R, Nathan J. Effect of forecast errors on rolling horizon master production schedule cost performance for various replanning intervals. Production Planning and Control 1999;10(7):682–9.
[35] Tang O, Grubbström RW. Planning and replanning the master production schedule under demand uncertainty. International Journal of Production Economics 2002;78(3):323–34.
[36] Xie J, Lee TS, Zhao X. Impact of forecasting errors on the performance of capacitated multi-item production systems. Computers & Industrial Engineering 2004;46(2):205–19.
[37] Brandimarte P. Multi-item capacitated lot-sizing with demand uncertainty. International Journal of Production Research 2006;44(15):2997–3022.
[38] Huang M-G, Chang P-L, Chous Y-C. Demand forecasting and smoothing capacity planning for products with high random demand volatility. International Journal of Production Research 2007;46(12):3223–39.
[39] Robinson EP, Sahin F, Gao L-L. Master production schedule time interval strategies in make-to-order supply chains. International Journal of Production Research 2007;46(7):1933–54.
[40] Leung SCH, Lai KK, Ng W-L, Wu Y. A robust optimization model for production planning of perishable products. Journal of the Operational Research Society 2007;58(4):413–22.