A Capacitated Hierarchical Approach to Make–to–Order Production

Klaus Neumann, Christoph Schwindt

Institut für Wirtschaftstheorie und Operations Research,
University of Karlsruhe,
D–76128 Karlsruhe, Germany
E–mail: {neumann, schwindt}@wior.uni-karlsruhe.de

ABSTRACT. *We propose a four–stage capacity–oriented hierarchical approach to make–to–order production. The planning stages of capacitated master production scheduling, multi–level lot sizing, temporal plus capacity planning, and shop floor scheduling are considered. We formulate the optimization problems arising at the individual stages and briefly discuss appropriate solution methods. In particular, we devise an integrated approach to capacitated lot sizing and temporal plus capacity planning of multi–level products.*

RÉSUMÉ. *Nous proposons une approche hiérarchique à quatre niveaux pour l'ordonnancement d'une production sur commande. A chaque niveau les effectifs de l'entreprise sont pris en considération. Nous traitons la plannification de l'assortissement à moyen terme, la gestion des stocks, l'ordonnancement des effectifs ainsi que l'ordonnancement des ateliers à court terme. A chaque niveau, le problème d'optimisation est formulé et des algorithmes appropriés sont présentés. En particulier, nous proposons une approche intégrée à la gestion des stocks et l'ordonnancement à contraintes de ressources de la production de produits composés.*

KEY WORDS*: Make–to–order production, hierarchical planning, project scheduling, shop floor scheduling, lot sizing*

MOTS–CLÉS*: Production sur commande, plannification hiérarchique, ordonnancement de projets, ordonnancements d'ateliers, gestion des stocks*

1. Introduction

As the pace in which technology, products, and markets are changing increases, rapid change also occurs throughout all phases of production. If all products are manufactured in response to firm customer orders, that is, no inventories are built up for future sale, we speak of *make–to–order production.* In addition to given customer orders, prescribed delivery dates for these orders have to be met. Make–to–order production is typical of single–item and small–batch production. It is well–known that make–to–order production becomes more and more important in practice, whereas mass production for the market is decreasing.

Several approaches to hierarchical production planning have been discussed in literature (see, for example, Hax & Meal 1975, Drexl et al. 1994, Steven 1994, Schneeweiss 1995, Carravilla & de Sousa 1995, and Franck et al. 1997). Most models, however, neither directly refer to make–to–order production nor observe the limited availability of resources at all production levels. In what follows, we propose a capacity–oriented hierarchical approach to production planning tailor–made for make–to–order production, which consists of the four planning stages

- Capacitated master production scheduling
- Multi–level lot sizing
- Temporal plus capacity planning
- Shop floor scheduling

and where the degree of aggregation of products and resources decreases from stage to stage. We model the optimization problems arising at the individual planning stages and briefly sketch how to solve those problems.

2. Capacitated master production scheduling

At the first planning stage, a capacitated *master production schedule* (abbreviated *MPS*) has to be determined. An MPS specifies the exact amounts and timing of production of each final product and some main components (together called *main products*) such that the resource requirements of work centers or main branches are as constant and as low as possible over time. The latter intention of keeping the workload low and constant over time seems to be best for providing good feasible solutions at all subsequent planning stages. The planning horizon of the first stage is usually about one year comprising twelve periods of one month each.

For the final products, the amounts to produce are given by the customer orders. Also, month–precise delivery dates for the final products are given and must be met. From the order quantities of final products and the product structure of the company in consideration (given by bills of materials or a gozinto graph), the gross requirements for main components at lower production levels can be computed. This requires the solution of a system of linear equations and is known as bills of materials explosion (cf. Nahmias 1997 and Neumann 1996).

To find the times when to produce the main products, we model the problem of determining a capacitated MPS as a *resource–levelling project scheduling problem* keeping in mind that we want to have a nearly constant workload over time. To do so, we first construct a *single–project network* with resource constraints for each individual customer order. The manufacture or assembly of the gross requirement for each main product i of such a customer order is regarded as an activity i of a project. The model used for the project is an activity–on–node network, that is, each activity i is assigned to a node i of the network. The duration or execution time D_i of activity i results from summing up the setup and processing time of product i itself and of the main components of product i at lower production levels. To obtain the minimum time lag d_{ik}^{min} between the start of activity i and the start of any subsequent activity k, some transition time (the sum of conveyance time and some waiting time) generally has to be added to D_i. This transition time is often estimated to be some surcharge for transportation and handling. Sometimes, a maximum time lag d_{ik}^{max} between the start of activities i and k may be given in addition, for example, resulting from delivery dates or maximum reaction time of unstable chemical substances. A minimum time lag d_{ik}^{min} is modelled by an arc $\langle i,k\rangle$ with weight $b_{ik} := d_{ik}^{min}$ in the project network, and a maximum time lag d_{ik}^{max} by an arc $\langle k,i\rangle$ with weight $b_{ki} := -d_{ik}^{max}$. How to construct an activity–on–node network in that way, is discussed in more detail in Franck & Neumann (1997) and Neumann & Schwindt (1997).

The resources required for carrying out the activities of the project are determined by summing up the respective machine units and workers needed. To avoid peak demand for resources, the resource requirements for product i are assumed to be distributed uniformly over the execution time D_i of activity i so that the resource demand rates are constant.

The single–project networks for all customer orders are then joined together to make a *multi–project network*. We add a supersource α , a supersink ω, and arcs from α to all sources of the single–project networks and from all sinks of the single–project networks to ω, with appropriate weights. A delivery date or deadline $\overline{d}_i$ for some product i can be modelled by a maximum time lag of size $\overline{d}_i - D_i$ between the dummy activity corresponding to supersource α and the start of activity i.

Next, we want to formulate the *resource–levelling problem* for the multi–project network. Let $1, ..., n$ be the real activities or nodes, respectively, of the project and let the fictitious activities 0 and $n+1$ correspond to the supersource α and supersink ω, respectively. Thus, $V = \{0, 1, ..., n+1\}$ is the set of activities or nodes, respectively. Let E be the set of the arcs of the network.

Let $\mathcal{K}$ be the set of (renewable) resources, say, machines and workers, and let $R_\kappa > 0$ be the capacity of resource $\kappa \in \mathcal{K}$ available and $r_{i\kappa}$ be the amount of resource κ used for processing activity $i \in V$ where $0 \leq r_{i\kappa} \leq R_\kappa$ and $r_{0\kappa} := r_{n+1,\kappa} := 0$. Moreover, let S_i be the start time of activity $i \in V$ where $S_0 := 0$. Given a *schedule* $\boldsymbol{S} = (S_0, S_1, \ldots, S_{n+1})$

$$\mathcal{A}^S(t) := \{i \in V | S_i \leq t < S_i + D_i\}$$

is called the *active set* (i.e. the set of activities in progress) at time t (or in time interval $[t, t+1[$ or period $t+1$, respectively) with $t = 0, 1, ..., \overline{d}-1$, where $\overline{d}$ is the given planning horizon, and

$$r_\kappa^S(t) := \sum_{i \in \mathcal{A}^S(t)} r_{i\kappa}$$

is the amount of resource κ used at time t. The resource–levelling problem is then as follows:

$$\begin{array}{lllr}
\text{Minimize} & F(S_0, \ldots, S_{n+1}) & & [1] \\
\text{subject to} & r_\kappa^S(t) \leq R_\kappa & (\kappa \in \mathcal{K}, t = 0, \ldots, \overline{d}-1) & [2] \\
& S_k - S_i \geq b_{ik} & (\langle i, k \rangle \in E) & [3] \\
& S_0 = 0, \;\; S_{n+1} = \overline{d} & & [4] \\
& S_i \in \mathbb{Z}_{\geq 0} & (i \in V) & [5]
\end{array}$$

Inequalities [2] represent the resource constraints and relations [3], [4], and [5] the time constraints. Possible objective functions F in [1] are, for example,

$$\sum_{\kappa \in \mathcal{K}} g_\kappa \sum_{t=0}^{\overline{d}-1} |r_\kappa^S(t) - \overline{R}_\kappa| \qquad [6]$$

$$\sum_{\kappa \in \mathcal{K}} g_\kappa \sum_{t=0}^{\overline{d}-1} [r_\kappa^S(t) - \overline{r}_\kappa]^2 \qquad [7]$$

$$\sum_{\kappa \in \mathcal{K}} g_\kappa \max_{t=0,\ldots,\overline{d}-1} r_\kappa^S(t) \qquad [8]$$

where $g_\kappa > 0$ is a weighting factor (e.g. the cost per unit of resource κ used), $\overline{R}_\kappa$ is some target value for the usage of resource κ, and $\overline{r}_\kappa := \sum_{i \in V} r_{i\kappa} D_i / \overline{d}$ is the average resource utilization. Objective functions [6] and [7] express the deviation of the consumption of resources from a target value for the usage and the average utilization, respectively, of the resources. Objective function [8] represents the sum of the maximal resource costs per period and is expedient if very expensive resources have to be purchased to meet the customer orders.

A solution $S_0, ..., S_{n+1}$ to problem [1] to [5] provides month–precise milestones for the production of the gross requirements for the main products. The corresponding quantities $r_\kappa^S(t)$ represent the resource requirements. Pseudopolynomial heuristic procedures for the resource–levelling problem without resource constraints [2] have been proposed for the first time by Brinkmann & Neumann (1996), which have been tested for problem instances with up to 100 activities and several resources. Neumann & Zimmermann (1997) have devised polynomial heuristics for the resource–levelling problem with resource constraints [2], which approximately solve problem instances with 500 activities and several resources within reasonable computing time.

3. Multi–level lot sizing

At the stage of *lot sizing* (*LS stage*, for short) the main products are decomposed into *intermediate products* (final products and subassemblies) for which week–precise production quantities (also called *lots* or *production orders*) are computed. The gross requirements for the main products are given by the MPS. The planning horizon of roughly three months comprises $T = 13$ periods of one week each.

A final product is generally made up of several subassemblies. The product structure is assumed to be general, that is, the corresponding gozinto graph represents an arbitrary acyclic digraph. The production of the subassemblies and final products requires several resources. Each resource corresponds to a group of uniform machines. The processing of a product on a resource incurs setup cost, setup time, and processing time. Setup and processing times are given in time units (for example, hours). For a given resource, the (aggregated) per–period availability corresponds to the workload in time units which can be done by the uniform machines of the corresponding group within one week. The objective is to determine lots for the intermediate products such that no backlogging occurs, the per–period availabilities of all resources are observed in all periods, and the sum of setup and inventory holding costs is minimized. This problem can be formulated as a *multi–level capacitated lot sizing problem MLCLSP*.

Let $\mathcal{P}$ be the set of (intermediate) products and $\mathcal{K}$ be the set of resources. For each $j \in \mathcal{P}$, an inventory holding cost h_j per period and unit of product j and a setup cost s_j are given. The input coefficient a_{jl} corresponds to the number of units of product j which are directly installed into one unit of product $l \in \mathcal{S}_j$, where $\mathcal{S}_j$ is the set of direct successors of product j in the product structure. $d_{j\tau}$ denotes the gross requirements of product j in period τ ($\tau = 1, \dots, T$). $\vartheta_{j\kappa}$ and $p_{j\kappa}$ correspond to the setup and the processing workloads of product j on resource κ, respectively. For each resource $\kappa \in \mathcal{K}$ the per–period availability $A_{\kappa\tau}$ has to be observed. We introduce three types of decision variables. The lot size of product j in period τ is denoted by $q_{j\tau}$. $y_{j\tau}$ represents the stock of product j at the end of period τ. Finally, the binary variables $\gamma_{j\tau}$ are defined to be 1 if the lot size of product j in period τ is positive, and 0 otherwise.

The multi–level capacitated lot sizing problem can be formulated as follows:

$$\text{Minimize} \quad \sum_{\tau=1}^{T} \sum_{j \in \mathcal{P}} (h_j y_{j\tau} + s_j \gamma_{j\tau}) \qquad [9]$$

$$\text{subject to} \; y_{j,\tau-1} + q_{j\tau} - \sum_{l \in \mathcal{S}_j} a_{jl} q_{l\tau} - y_{j\tau} = d_{j\tau} \quad (j \in \mathcal{P}, \tau = 1, \ldots, T) \qquad [10]$$

$$\sum_{j \in \mathcal{P}} (\vartheta_{j\kappa} \gamma_{j\tau} + p_{j\kappa} q_{j\tau}) \leq A_{\kappa\tau} \quad (\kappa \in \mathcal{K}, \tau = 1, \ldots, T) \qquad [11]$$

$$q_{j\tau} - M\gamma_{j\tau} \leq 0 \quad (j \in \mathcal{P}, \tau = 1, \ldots, T) \qquad [12]$$

$$y_{j0} = 0 \quad (j \in \mathcal{P}) \qquad [13]$$

$$q_{j\tau} \geq 0, \; y_{j\tau} \geq 0 \quad (j \in \mathcal{P}, \tau = 1, \ldots, T) \qquad [14]$$

$$\gamma_{j\tau} \in \{0, 1\} \quad (j \in \mathcal{P}, \tau = 1, \ldots, T) \qquad [15]$$

The objective function in [9] is the sum of setup and inventory holding costs. Relations [10] represent the inventory balance equations which prevent backlogging. By inequalities [11] the limited per–period resource availabilities are observed. Inequalities [12] with $M := \max_{j \in \mathcal{P}} \sum_{\tau=1}^{T} d_{j\tau}$ ensure that the binary setup variable $\gamma_{j\tau}$ equals 1 exactly if there is a corresponding lot of size $q_{j\tau} > 0$. By [13] we initialize the initial stocks with zero. Constraints [14] and [15] define the decision variables to be nonnegative reals and binaries, respectively.

Due to setup times and resource constraints the feasibility problem of MLCLSP is $\mathcal{NP}$–complete. The most advanced heuristic for MLCLSP is the Lagrangean–based approach of Tempelmeier & Derstroff (1993, 1996). By relaxing the inventory balance equations [10] and capacity constraints [11], a decomposition of the original problem into several *single–level uncapacitated lot sizing problems SLULSP* is obtained, which can be solved to optimality in time $\mathcal{O}(|\mathcal{P}|)$ by dynamic programming. Violations of the relaxed constraints are taken into consideration by a linear penalty function whose Lagrangean multipliers are updated by a subgradient optimization procedure. A lower bound on the minimum objective function value can be derived from the optimal solutions of the SLULSP instances. Upper bounds are obtained by the generation of feasible solutions. A solution for which the inventory balance equations are observed can be computed as follows. First, the SLULSP instances corresponding to final products are solved. From the resulting lot sizes, the secondary requirements of products at the immediate predecessor level in the product structure are generated by a bills of materials explosion. These secondary requirements represent the demand of the SLULSP instances of products at that predecessor level. In the same way we proceed. A feasible solution satisfying the capacity constraints is constructed by a so–called heuristic scheduling procedure which tries to shift workload from periods with capacity overloads to periods with positive left–over availabilities. This is done until a feasible solution has been determined or a prescribed maximum number of passes has been attained.

4. Temporal plus capacity planning

Intermediate products may be further decomposed into *individual products* (final products, subassemblies, and main components). At the stage of *temporal plus capacity planning* (abbreviated *TCP*), shift–precise production orders for all individual products have to be determined for each week (period of the LS stage) observing the limited resources (groups of uniform machines).

In what follows, the processing of an individual product will be referred to as a *job*. A job can be represented by a sequence of *tasks* where a task $T_{\kappa j}$ corresponds to the processing of an individual product j on resource $\kappa \in \mathcal{K}$. For each individual product j the task sequence $(T_{\alpha_j j}, \ldots, T_{\omega_j j})$ is assumed to be given by process plans. α_j and ω_j denote the first and last group of uniform machines in the task sequence of product j, respectively.

The weekly gross requirements for the individual products can be found by a bills of materials explosion from the production orders for intermediate products computed at the LS stage. Since all shift–precise production orders have to be carried out within one week, we aim at minimizing the maximal completion time of all jobs, i.e. the *makespan*.

For a period τ of the LS stage, cumulative workload constraints

$$\sum_{j \in \mathcal{P}} (\vartheta_{j\kappa}\gamma_{j\tau} + p_{j\kappa}q_{j\tau}) \leq A_{\kappa\tau}$$

have been observed. Resource requirements as well as resource availabilities have been measured in time units. Up to now, we have not taken into account that, in general, we have to cope with noncumulative capacity constraints. Since the number of individual uniform machines which are aggregated to a resource is limited, the maximal number of resource units available at any point in time is limited, too. At the LS stage, say m uniform machines of a shop are aggregated to one resource κ with availability $A_{\kappa\tau} = m\Delta_{LS}$, where Δ_{LS} represents the length of a lot sizing period in time units. At the TCP stage, the corresponding resource has a capacity of $R_\kappa = m$, since at any point in time, items can be processed on at most m machines in parallel.

Obviously, the limited capacity of resources may give rise to waiting times which cannot be taken into consideration by cumulative workload constraints. In particular, these waiting times depend on the manner how resource conflicts between competing tasks are resolved.

The problem of scheduling tasks subject to resource capacity constraints such that the makespan is minimized can be modelled as a resource–constrained project scheduling problem. A given set $V = \{0, 1, \ldots, n, n+1\}$ of activities (corresponding to the start of the project, the tasks $T_{\kappa j}$ which have to be executed during the current period τ of the LS stage, and the completion of the project) have to be scheduled such that time constraints between activities are observed and the resource requirements for the activities do not exceed the capacity of any resource at any point in time. Each activity $i \in V$ is associated with a node in an activity–on–node project network. An arc $\langle i, k \rangle$

from activity i to activity k weighted by b_{ik} represents a minimum time lag d_{ik}^{min} of b_{ik} units of time between the start of activities i and k. In case of $b_{ik} < 0$, the negative minimum time lag b_{ik} can be interpreted as a maximum time lag d_{ki}^{max} between the starts of activities k and i. By E we again denote the set of arcs of the project network. During its duration D_i, activity i uses $r_{i\kappa}$ units of the capacity R_κ of resource $\kappa \in \mathcal{K}$. Let S_i be the start time of activity i and $\boldsymbol{S} = (S_0, \ldots, S_{n+1})$ be a schedule. As in Section 2, let $\mathcal{A}^S(t)$ be the active set and $r_\kappa^S(t)$ be the usage of resource κ at time t. With $\overline{T}$ being an upper bound on the minimal project duration, the *resource–constrained project scheduling problem with minimum and maximum time lags RCPSP/max* can be formulated as follows:

$$\begin{array}{lllr}
\text{Minimize} & S_{n+1} & & [16] \\
\text{subject to} & r_\kappa^S(t) \le R_\kappa & (\kappa \in \mathcal{K}, t = 0, \ldots, \overline{T} - 1) & [17] \\
& S_k - S_i \ge b_{ik} & (\langle i, k \rangle \in E) & [18] \\
& S_i \in \mathbb{Z}_{\ge 0} & (i \in V) & [19]
\end{array}$$

Objective function S_{n+1} in [16] represents the project duration. Inequalities [17] secure feasibility with respect to the capacity constraints. Inequalities [18] and [19] represent the time constraints.

In the following, we will sketch how to generate an appropriate RCPSP/max instance for the scheduling of production orders of a period τ at the lot sizing stage.

Let $\overline{\mathcal{P}} \supseteq \mathcal{P}$ be the set of individual products. First, we determine production quantities $q_{j\tau}$ for products $j \in \overline{\mathcal{P}} \setminus \mathcal{P}$ by a bills of materials explosion. Let $\overline{\mathcal{P}}_\tau := \{j \in \overline{\mathcal{P}} | q_{j\tau} > 0\}$ be the set of products j produced in period τ. By definition, a task uses a constant amount of resources during its processing. If, for instance, product j has to be processed subsequently on several groups $\kappa \in \mathcal{K}$ of uniform machines, a task corresponds to the processing of all $q_{j\tau}$ units of j on one of these resources κ. Let $m_{T_{\kappa j}} > 0$ be the number of units of resource κ required for the processing of task $T_{\kappa j}$. The setup time $\vartheta_{T_{\kappa j}}$ of task $T_{\kappa j}$ corresponds to the setup workload per resource unit $\vartheta_{j\kappa}/m_{T_{\kappa j}}$ of product j on resource κ. By $p_{T_{\kappa j}}$ we denote the processing workload per resource unit $p_{j\kappa}/m_{T_{\kappa j}}$ of product j on resource κ. For the duration $D_{T_{\kappa j}}$ of task $T_{\kappa j}$ we obtain

$$D_{T_{\kappa j}} = \vartheta_{T_{\kappa j}} + q_{j\tau} p_{T_{\kappa j}}$$

Let $i = T_{\kappa j}$ and $k = T_{\lambda j}$ be two subsequent tasks in the task sequence of product $j \in \overline{\mathcal{P}}_\tau$. The execution of task k can start after the completion of the first unit of product j in task i. If $p_k < p_i$, however, task k would be interrupted after each completion of a unit of product j. Thus, we determine a minimum time lag b_{ik} between the start of tasks i and k as follows:

$$b_{ik} := \begin{cases} \vartheta_i + p_i - \vartheta_k, \text{ if } p_i \leq p_k \\ \vartheta_i + q_{j\tau} p_i - (q_{j\tau} - 1)p_k - \vartheta_k, \text{ otherwise} \end{cases}$$

Now, let $i = T_{\omega_j j}$ and $k = T_{\alpha_l l}$ be the last task in the task sequence of product j and the first task in the task sequence of a product $l \in \mathcal{S}_j$, respectively. By $AS(j) = (l_1, \ldots, l_{n_j})$ we denote the so–called *assignment sequence* of product j ($l_\mu \in \mathcal{S}_j$ for all $\mu = 1, \ldots, n_j$). $AS(j)$ determines the sequence in which completed units of product j are assigned to products l_μ at the immediate successor level in the product structure. The processing of the first unit of product j which will be available for the assembly of product $l_{\mu+1}$ is started after the completion of the last unit of product j dedicated to the assembly of product l_μ. Heuristics for the generation of appropriate assignment sequences can be found in Neumann & Schwindt (1997). Let $l = l_\nu$ be the ν-th product in $AS(j)$. For the minimum time lag b_{ik} between the starts of i and k we obtain

$$b_{ik} := \begin{cases} \vartheta_i + \sum\limits_{\mu=1}^{\nu-1} a_{jl_\mu} q_{l_\mu \tau} p_i + a_{jl} p_i - \vartheta_k, \text{ if } a_{jl} p_i \leq p_k \\ \vartheta_i + \sum\limits_{\mu=1}^{\nu} a_{jl_\mu} q_{l_\mu \tau} p_i - (q_{l\tau} - 1)p_k - \vartheta_k, \text{ otherwise} \end{cases}$$

The resource capacities R_κ can be derived from the availabilities $A_{\kappa\tau}$ of the lot sizing stage as follows:

$$R_\kappa := A_{\kappa\tau} / \Delta_{LS}$$

The renewable resource requirements $r_{i\kappa}$ represent the number of units m_i used for the processing of task $i = T_{\kappa j}$:

$$r_{i\kappa} := m_{T_{\kappa j}}$$

Finally, we introduce a maximum time lag of Δ_{LS} units of time between the initial task 0 and the completion task $n+1$ of the project. If the RCPSP/max instance obtained by the calculation of activity durations, minimum and maximum time lags, resource capacities, and resource requirements is solvable, there is a task schedule $\boldsymbol{S}^\tau$ for the production orders of period τ such that all tasks are completed within one lot sizing period, that is, $S^\tau_{n+1} \leq \Delta_{LS}$. Let Δ_{TCP} denote the length of a shift in time units (for example, eight hours). The production orders of the σ–th shift in the current period τ for individual products $j \in \overline{\mathcal{P}}_\tau$ correspond to the jobs with final tasks $i = T_{\omega_j j}$ for which it holds that

$$(\sigma - 1)\Delta_{TCP} < S^\tau_i + D_i \leq \sigma \Delta_{TCP}$$

RCPSP/max can be solved by a (truncated) branch–and–bound algorithm by Schwindt (1998). In each step, this algorithm solves optimization problems for which the resource constraints [17] have been relaxed. To take account of the limited resource capacities, additional *disjunctive precedence constraints*

$$\min_{k\in V_2} S_j \geq \min_{i\in V_1}(S_i + D_i) \qquad [20]$$

between nonempty sets of activities V_1 and V_2 are introduced instead.

Problem [16], [18], and [19] subject to additional disjunctive precedence constraints [20] can be solved to optimality by a pseudo–polynomial fix–point algorithm. Now let us assume that for the resulting schedule $\boldsymbol{S}^+$, there is an active set $\mathcal{A}^{S^+}(t)$ such that due to the resource constraints not all activities $i \in \mathcal{A}^{S^+}(t)$ can be processed simultaneously. We then examine all possibilities of delaying the minimum number of activities $k \in \mathcal{A}^{S^+}(t)$ which are necessary to resolve the resource conflict. The delayed activities are shifted to the point in time where the first activity $i \in \mathcal{A}^{S^+}(t)$ which is not delayed has been completed. More precisely, we enumerate all partitions $\{V_1, V_2\}$ of $\mathcal{A}^{S^+}(t)$ such that V_1 represents an inclusion–maximal set of activities which may be processed at the same time. The selection of a partition $\{V_1, V_2\}$ corresponds to the introduction of additional disjunctive precedence constraints as given by [20]. A feasible schedule has been determined if the optimal solution $\boldsymbol{S}^+$ to [16], [18], and [19] subject to the additional constraints of type [20] is resource–feasible.

A preprocessing algorithm, several fathoming rules, and tight lower bounds allow for a substantial reduction in size of the branch and bound tree (cf. Schwindt 1998).

5. Integrated Lot Sizing and Temporal plus Capacity Planning

Let $\boldsymbol{S}^\tau = (S_0^\tau, \ldots, S_{n+1}^\tau)$ be the schedule generated at the TCP stage for period τ of the lot sizing problem. If the completion time $S_i^\tau + D_i$ of a task i exceeds the given deadline Δ_{LS}, $\boldsymbol{S}^\tau$ does not represent a feasible schedule for the processing of the production orders of products $j \in \overline{\mathcal{P}}_\tau$. In that case, we have to determine new lot sizes for periods $1, \ldots, T$. The lot size of at least one product $j \in \overline{\mathcal{P}}_\tau$ has to be reduced. This can be done by reduction of the availabilities of resources whose capacity has been violated. Let $\overline{A}_{\kappa\tau} := \max_{i\in V: r_{i\kappa}>0}(S_i^\tau + D_i)$ be the completion time of the last task which requires resource κ. If $\overline{A}_{\kappa\tau} > \Delta_{LS}$, we have to decrease the availability of resource κ in period τ for the next run of the lot sizing step as follows:

$$A_{\kappa\tau} := \min\left\{(1-\alpha\frac{\overline{A}_{\kappa\tau} - \Delta_{LS}}{\overline{A}_{\kappa\tau}})A_{\kappa\tau}, \sum_{j\in\overline{\mathcal{P}}_\tau}(\vartheta_{j\kappa} + p_{j\kappa}q_{j\tau}) - \epsilon\right\} \text{ with } \epsilon > 0$$

The original availability $A_{\kappa\tau}$ is diminished proportionally to the relative deadline violation and a control parameter $\alpha \in (0, 1]$. Now, let us assume that the modified availability is still sufficient to process the same production orders at the LS stage (that is, $A_{\kappa\tau} \geq \sum_{j\in\overline{\mathcal{P}}_\tau}(\vartheta_{j\kappa} + p_{j\kappa}q_{j\tau})$). In this case, we enforce the reduction of at least one lot size in period τ. A similar approach for the special case of a job shop environment can be found in Lambrecht & Vanderveken (1979).

An implicit assumption of the inventory balance equations is that each product l can be processed at the same time as its predecessor products j in the product structure. Since the assembly of product l can only be started after the completion of at least a_{jl} units of product j, we introduce a lead time $z(j,\tau)$ for the production order of product j. Thus, relation [10] in problem MLCLSP is replaced by $y_{j,\tau-1} + q_{j\tau - z(j,\tau)} - \sum_{l \in \mathcal{S}_j} a_{jl} q_{l\tau} - y_{j\tau} = d_{j\tau}$. For the determination of appropriate lead times we have to take into account that the above reduction of resource availabilities leads to smaller lot sizes in the next run of the lot sizing heuristic:

$$z(j,\tau) := \left\lfloor \frac{\max_{\kappa \in \mathcal{K}} \overline{A}_{\kappa\tau} - S^{\tau}_{T_{\alpha_j j}}}{\Delta_{LS}} \left(1 - \alpha \max_{\kappa \in \mathcal{K}} \frac{\overline{A}_{\kappa\tau} - \Delta_{LS}}{\overline{A}_{\kappa\tau}} \right) \right\rfloor$$

Figure 1 summarizes the algorithm for integrated lot sizing and temporal plus capacity planning.

```
Solve MLCLSP by algorithm of Tempelmeier & Derstroff
IF no feasible solution is found THEN STOP
$\tau := 0$
WHILE $\tau \leq T$
    Generate RCPSP/max for period $\tau$
    Solve RCPSP/max by algorithm of Schwindt
    IF $A_{\kappa\tau} \leq \Delta_{LS}\ \forall \kappa \in \mathcal{K}$ THEN $\tau := \tau + 1$
    ELSE
        Update resource availabilities $A_{\kappa\tau}$
        Determine lead times $z(j,\tau)$
        Solve MLCLSP by algorithm of Tempelmeier & Derstroff
        IF no feasible solution is found THEN STOP
        $\tau := 0$
    END (*IF*)
END (*WHILE*)
```

Figure 1. *Integrated lot sizing and temporal plus capacity planning*

6. Shop floor scheduling

The final planning stage is concerned with processing the jobs on the individual machines so that their due dates are met. The due date of a job is defined to be the completion time of the respective production order determined at the TCP stage. The time horizon for shop floor scheduling is often one working day and time units (length of periods) are typically anywhere from 5 to 60 minutes.

Let $\mathcal{J}$ be the set of all jobs, and let δ_j be the due date and C_j be the completion time of job $j \in \mathcal{J}$. Then $\varepsilon_j(\delta_j - C_j)$ is an earliness cost when job j is completed early and $\tau_j(C_j - \delta_j)$ is a tardiness cost when job j is completed late, where $\varepsilon_j \geq 0$ and $\tau_j \geq 0$ are the earliness and tardiness cost of job j, respectively, per unit time.

$$h_j(C_j) := \varepsilon_j(\delta_j - C_j)^+ + \tau_j(C_j - \delta_j)^+$$

where $(z)^+ := \max(z, 0)$ represents a penalty cost for job j. Possible objective functions to be minimized are

(a) $\sum\limits_{j \in \mathcal{J}} h_j(C_j)$,

(b) $\max\limits_{j \in \mathcal{J}} h_j(C_j)$, or

(c) a linear combination of such a sum– and max–penalty cost.

The shop floor problem to be solved consists of two parts. In general, several uniform machines of the same type are available which differ in speed. Thus at first, if a job $j \in \mathcal{J}$ has to be processed on a certain group of uniform machines (also called *machine type*) $\kappa \in \mathcal{K}$, it has to be assigned to an individual machine of that type. Recall that the processing of job j on machine type κ corresponds to a *task* $T_{\kappa j}$. The execution of job j on an individual machine M_κ of type κ is termed an *operation* $o_{\kappa j}$. The different operations or respectively uniform machines of the same type belonging to a task are viewed as *modes*, and the problem to be solved is called a *mode assignment problem.* Given a task sequence for each job, a (feasible) solution to the mode assignment problem provides an operation sequence for the corresponding job. Solution procedures for the mode assignment problem (for example, a heuristic method where for each task, the corresponding modes are ordered according to nondecreasing processing times) have been proposed by Serafini & Speranza (1994) and Kolisch (1995).

Secondly, given an operation sequence for each job, an earliness–tardiness job shop problem with one of the objective functions (a), (b), or (c) has to be solved. We briefly want to sketch a *two–step approach* to approximately solving that job shop problem.

In *step 1*, we apply the well–known priority–rule method by Giffler and Thompson (cf. Giffler & Thompson 1960 or Neumann 1996) for the job shop problem with an appropriate priority rule, which takes early and late completion of jobs into account. For each eligible operation $o_{\kappa j}$ (i.e. all predecessors of $o_{\kappa j}$ have already been scheduled) of a job j, we determine an *optimistic estimate* $OC_{\kappa j}$ and a *pessimistic estimate* $PC_{\kappa j}$ of the completion time C_j of job j. For the optimistic estimate $OC_{\kappa j}$, we assume that the remaining operations of job j are the first to be scheduled on the respective machines, and for the pessimistic estimate $PC_{\kappa j}$, the remaining operations of job j are the last to be scheduled. To determine an estimate $PC_{\kappa j}$ for job j, the sequences of the jobs $j' \neq j$ on all machines have to be known. The problem of finding optimal job sequences, however, is as hard as the original job shop problem. Thus, we only schedule the unscheduled operations of the jobs j' in a simple way, for example, job after job observing the operation sequence for each job, and carry out some local left–shifts.

For the objective functions (a), (b), or (c) it is then expedient to choose the priority value

$$\nu_{\kappa j} := \tau_j(PC_{\kappa j} - \delta_j)^+ - \varepsilon_j(\delta_j - OC_{\kappa j})^+ \qquad [21]$$

for operation $o_{\kappa j}$. That is, the smaller the due date δ_j for job j is in comparison with $OC_{\kappa j}$ and $PC_{\kappa j}$ the greater is the priority value $\nu_{\kappa j}$. If eligible operations $o_{\kappa j}$ are scheduled according to nonincreasing values $\nu_{\kappa j}$, then late jobs are preferred to early jobs. Moreover, the Giffler–Thompson heuristic tries to schedule an eligible operation each time a machine is freed, which is appropriate for minimizing makespan but not necessarily for minimizing earliness–tardiness objective functions. Thus, some additional right–shifting of jobs seems be reasonable which will be done in step 2.

In *step 2* of the approach, given the schedule found in step 1, the operations are locally right–shifted so that the earliness–tardiness objective function (a), (b), or (c) is minimized. Let job j_κ be that job which is processed immediately after job j on machine M_κ in the schedule computed in step 1, and let $v_{jj_\kappa} \geq 0$ be the idle time on machine M_κ between the completion of operation $o_{\kappa j}$ and the start of operation $o_{\kappa j_\kappa}$. Moreover, let M_{κ_j} be that machine on which job j is processed immediately after processing on machine M_κ, and let $w_{\kappa\kappa_j} \geq 0$ be the waiting time of job j between the completion of operation $o_{\kappa j}$ and the start of operation $o_{\kappa_j j}$. If each operation $o_{\kappa j}$ is right–shifted $s_{\kappa j} \geq 0$ time units, it must hold that

$$s_{\kappa j} - s_{\kappa j_\kappa} \leq v_{jj_\kappa} \qquad (M_\kappa \in \mathcal{M}_j, j \in \mathcal{J}) \qquad [22]$$

$$s_{\kappa j} - s_{\kappa_j j} \leq w_{\kappa\kappa_j} \qquad (M_\kappa \in \mathcal{M}_j, j \in \mathcal{J}) \qquad [23]$$

where $\mathcal{M}_j$ is the set of machines on which job j has to be processed. Let $M_{\omega_j} \in \mathcal{M}_j$ be the last machine on which job j is processed. Then the minimization of objective function (a) leads to the problem

$$\begin{array}{ll} \text{Minimize} & \sum_{j\in\mathcal{J}} [\varepsilon_j(\delta_j - C_j - s_{\omega_j j})^+ + \tau_j(C_j + s_{\omega_j j} - \delta_j)^+] \\ \text{subject to} & [22],\ [23] \\ & s_{\kappa j} \geq 0 \qquad (M_\kappa \in \mathcal{M}_j, j \in \mathcal{J}) \end{array}$$

To transform that optimization problem into a linear program, we introduce variables x_j and y_j, which represent the earliness and tardiness, respectively, of the right–shifted job j. Then we obtain the linear program

$$\begin{array}{lll} \text{Minimize} & \sum_{j\in\mathcal{J}} [\varepsilon_j x_j + \tau_j y_j] & \\ \text{subject to} & [22],\ [23] & \\ & x_j + s_{\omega_j j} \geq \delta_j - C_j & (j \in \mathcal{J}) \\ & y_j - s_{\omega_j j} \geq C_j - \delta_j & (j \in \mathcal{J}) \\ & s_{\kappa j} \geq 0 & (M_\kappa \in \mathcal{M}_j, j \in \mathcal{J}) \\ & x_j \geq 0,\ y_j \geq 0 & (j \in \mathcal{J}) \end{array}$$

Objective functions (b) and (c) can be dealt with analogously.

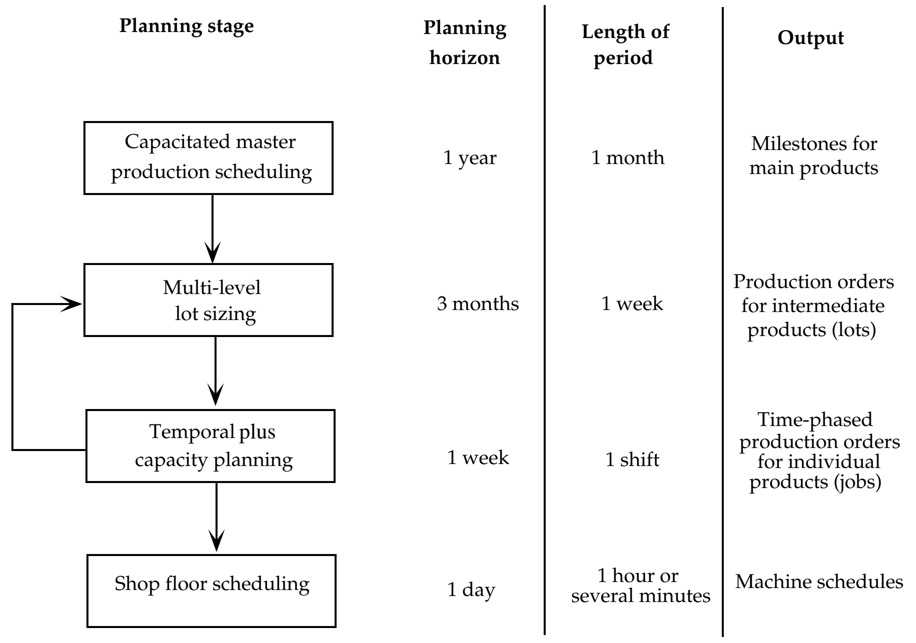
An *improvement of the solution procedure* for the shop floor scheduling problem is to include the mode assignment in step 1 of the job shop heuristic. That is, both the task to be scheduled next on a machine of a certain type and the individual machine of that type on which the operation is to be processed should be chosen using some appropriate priority rule. Let $\mathcal{M}$ be the set of all machines. For simplicity, we assume that each job has to be processed on each machine type exactly once. At the beginning we set $\mathcal{M}_j := \mathcal{M}$ for all $j \in \mathcal{J}$. Then in step 1, we apply the Giffler–Thompson heuristic as above using the priority values $\nu_{\kappa j}$ for eligible operations $o_{\kappa j}$ with $M_\kappa \in \mathcal{M}_j$ given by [21]. The operation $o_{\kappa j}$ to be scheduled next is always one with largest $\nu_{\kappa j}$. Once $o_{\kappa j}$ is scheduled, we eliminate all machines of the same type as M_κ from $\mathcal{M}_j$.

A different *three–step decomposition approach* for solving the shop floor scheduling problem has been proposed by Serafini & Speranza (1994). In step 1, a mode–assignment problem is solved by a sorting heuristic. In step 2, a sequencing problem (i.e. computing a job sequence for each machine) is solved by an appropriate shifting bottleneck procedure. In step 3, a scheduling problem (i.e. finding the start times of all operations) is modelled as a network flow problem and solved by the network simplex method. The three steps are linked by feedback and can be solved iteratively.

7. Conclusions

A capacity–oriented hierarchical approach to make–to–order production has been presented, which consists of four planning stages illustrated in Fig. 2. For the individual planning stages, we have formulated suitable optimization problems and briefly discussed appropriate solution methods. Recall that for the lot sizing and temporal plus capacity planning stages, the solution procedure requires alternating between the two stages until an appropriate feasible solution at the latter stage is found.

Important areas of future research are, for example, the development of more efficient solution methods for the multi–mode earliness–tardiness job shop problem. Moreover, similar hierarchical approaches to different types of production should be devised.



Planning stage
Planning horizon
Length of period
Output
Capacitated master production scheduling
1 year
1 month
Milestones for main products
Multi-level lot sizing
3 months
1 week
Production orders for intermediate products (lots)
Temporal plus capacity planning
1 week
1 shift
Time-phased production orders for individual products (jobs)
Shop floor scheduling
1 day
1 hour or several minutes
Machine schedules

Figure 2. *Overview of the planning stages*

Acknowledgement.
This research has been supported in part by the Deutsche Forschungsgemeinschaft (Grant Ne 137/4).

References

[BRI 96] Brinkmann, K., Neumann, K. (1996), Heuristic procedures for resource–constrained project scheduling with minimal and maximal time lags: the resource–levelling and minimum project–duration problems. *J. Decision Systems 5*, 129–155

[CAR 95] Carravilla, M.A., de Sousa, J.P. (1995), Hierarchical production planning in a make–to–order company: a case study. *EJOR 86*, 43–56

[DRE 94] Drexl, A., Fleischmann, B., Günther, H.–O., Stadtler, H., Tempelmeier, H. (1994), Konzeptionelle Grundlagen kapazitätsorientierter PPS–Systeme. *ZfbF 46*, 1022–1045

[FRA 97] Franck, B., Neumann, K. (1997), Resource–constrained project scheduling with time windows. *Report WIOR–492*, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe

[FRA 97a] Franck, B., Neumann, K., Schwindt, C. (1997), A capacity–oriented hierarchical approach to single–item and small–batch production planning using project scheduling methods. *Oper. Res. Spektrum 19*, 77–85

[GIF 60] Giffler, B., Thompson, G.L. (1960), Algorithms for solving production scheduling problems. *Oper. Res. 8*, 487–503

[HAX 75] Hax, A.C., Meal, H.C. (1975), Hierarchical integration of production planning and scheduling. In: Geisler, M. (Ed.): *Logistics*, TIMS Studies in Management Science, Vol. 1, Elsevier, Amsterdam, 53–69

[HER 96] Herroelen, W., Demeulemeester, E., De Reyck, B. (1996), Resource–constrained project scheduling – a survey of recent developments. *Invited Paper* at the Symposium on Operations Research at Braunschweig, Germany

[KOL 95] Kolisch, R. (1995), *Project Scheduling under Resource Constraints.* Physica, Heidelberg

[LAM 79] Lambrecht, M.R., Vanderveken, H. (1979), Production scheduling and sequencing for multi–stage prodction systems. *Oper. Res. Spektrum 1*, 103–114

[NAH 97] Nahmias, S. (1997), *Production and Operations Analysis.* Irwin, Homewood

[NEU 96] Neumann, K. (1996), *Produktions– und Operations–Management.* Springer, Berlin

[NEU 97] Neumann, K., Schwindt, C. (1997), Activity–on–node networks with minimal and maximal time lags and their application to make–to–order production. *Oper. Res. Spektrum 19*, 205–217

[NEU 97a] NEUMANN, K., Zimmermann, J. (1997), Resource levelling for projects with schedule–dependent time windows, *Report WIOR–508*, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe, submitted to *EJOR*

[SCH 95] SCHNEEWEISS, C. (1995), Hierarchical structures in organisations: a conceptual framework. *EJOR 86*, 4–31

[SCH 98] SCHWINDT, C. (1998), *Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern.* Shaker, Aachen

[SER 94] SERAFINI, P., Speranza, M.G. (1994), A decomposition approach for a resource constrained scheduling problem. *EJOR 75*, 112–135

[STE 94] STEVEN, M. (1994), *Hierarchische Produktionsplanung.* Physica, Heidelberg

[TEM 93] TEMPELMEIER, H., Derstroff, M. (1993), Mehrstufige Mehrprodukt–Losgrößenplanung bei beschränkten Ressourcen und genereller Erzeugnisstruktur, *Oper. Res. Spektrum 15*, 63–73

[TEM 96] TEMPELMEIER, H., Derstroff, M. (1996), A Lagrangean–based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Mgmt Sci. 42*, 738–757