Kimms, Alf

Multi-level, single-machine lot sizing and scheduling: With initial inventory

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 329

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Suggested Citation: Kimms, Alf (1993) : Multi-level, single-machine lot sizing and scheduling: With initial inventory, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 329, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/149826

Nr. 329

Multi-Level, Single-Machine Lot Sizing and Scheduling (with Initial Inventory)

A. Kimms

September 1993

Dipl.-Inform. Alf Kimms, Lehrstuhl für Produktion und Logistik, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24118 Kiel, Germany

Abstract: This paper presents a mixed-integer program for the dynamic lot sizing and scheduling problem in a multi-level, single-machine environment. It turns out that in contrast to single-level problems the integration of initial inventory is a crucial aspect if generality should not be lost. It is shown how problem instances can efficiently be solved to suboptimality by using a so-called randomized regret based heuristic.

Keywords: Production planning, lot sizing, scheduling, PLSP, heuristics, multi-level.

1 Introduction

Lot sizing and scheduling, i.e. deciding which quantities of what items have to be produced at what time to meet the (known or estimated) demand, are two of the most serious production planning problems. Hence, many researchers tried to find appropriate models and solution methods for such kind of problems. Some of the most well-known models cover the *capacitated lot sizing problem* (CLSP) [de Both et al. 1984, Günther 1987], the *discrete lot sizing and scheduling problem* (DLSP) [Dinkelbach 1964, Haehling von Lanzenauer 1970, Fleischmann 1990, Salomon 1991] and the *continuous setup lot sizing problem* (CSLP) [Bitran and Matsuo 1986, Karmarkar et al. 1987, de Matta and Guignard 1989], respectively. A more recent one is Haase's model for *proportional lot sizing and scheduling problems* (PLSP) [Drexl and Haase 1992, Haase 1993]. While the CLSP-models cope with lot sizing only, the other models deal with scheduling, i.e. the problem of sequencing production quantities, as well.

In the sequel we make certain assumptions about the problems to handle: We assume a finite time horizon divided into discrete periods with dynamic but deterministic external (independent) demands for items. No shortages are allowed, i.e. the production must meet the demands promptly. Items may consist of other items which are to be produced (with an item specific lead time) before the former items can be manufactured. The intermediate items may in turn cause internal (dependent) demand for other items and so on. We impose no other restriction on this multi-level product structure than being acyclic. Due to the observation that many managers spot one single machine as a critical resource, e.g. one single machine is a bottleneck slowing down the whole production process, we concentrate on single-machine problems where all items are to be produced on that sole machine. The capacity per period of this machine is assumed to be constrained (but not necessarily constant over time). The production of one item consumes an item specific amount of the capacity. To produce an item the machine has to be setup for this item. Every setup causes item specific setup costs. Setup times and sequence dependent setup costs are ignored. Items that are produced within a period without a demand for these items within the same period are stored in inventory to meet future demand. While inventory is assumed to be uncapacitated, holding an item in inventory causes item specific holding costs. Some items may already be stored in inventory before we start the production. The objective is to find a production plan that is feasible and cheap, i.e. the sum of setup and holding costs should be as low as possible.

In the following section we will present a mixed-integer PLSP-model for lot sizing and scheduling problems of the above kind. We chose a PLSP-model because DLSP- and CSLP-models are based on the

restriction that at most one item can be produced per period, whereas PLSP-models allow at most one changeover per period, i.e. up to two items can be produced per period. Apparently, an optimal solution of a PLSP-instance is at least as good as an optimal solution of a corresponding DLSP- or CSLP-instance with the same data set. To compute a solution of a problem instance without initial inventory, we adopt a randomized regret based heuristic which was successfully used for generating suboptimal solutions in the single-level case [Haase 1993]. The multi-level extension of this heuristic is described in a later section. Since integrating initial inventory turns out to be more complex than in the single-level case (where initial inventory can simply be ignored without loss of generality), we will describe the details concerning initial inventory as well.

2 A PLSP-Model for Multi-Level, Single-Machine Problems

The multi-level, single-machine lot sizing and scheduling problem (with initial inventory and) with at most one changeover per period can be modelled as a mixed-integer program as follows:

$$\min \sum_{t=1}^{T} \sum_{j=1}^{J} (s_j x_{jt} + h_j I_{jt}) \tag{1}$$

subject to

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt} - \sum_{i \in S(j)} (a_{ji} q_{it}) \qquad (j = 1 .. J, t = 1 .. T) \tag{2}$$

$$I_{jt} \geq \sum_{\tau=t+1}^{\min\{t+v_j, T\}} \sum_{i \in S(j)} (a_{ji} q_{i\tau}) \qquad (j = 1 .. J, t = 0 .. T-1) \tag{3}$$

$$\sum_{j=1}^{J} y_{jt} \leq 1 \qquad (t = 1 .. T) \tag{4}$$

$$B (y_{jt} + y_{j(t-1)}) - q_{jt} \geq 0 \qquad (j = 1 .. J, t = 1 .. T) \tag{5}$$

$$x_{jt} - y_{jt} + y_{j(t-1)} \geq 0 \qquad (j = 1 .. J, t = 1 .. T) \tag{6}$$

$$\sum_{j=1}^{J} (p_j q_{jt}) \leq C_t \qquad (t = 1 .. T) \tag{7}$$

$$y_{jt} \in \{ 0 , 1 \} \qquad (j = 1 .. J, t = 1 .. T) \tag{8}$$

$$I_{jt} \geq 0, q_{jt} \geq 0, x_{jt} \geq 0 \qquad (j = 1 .. J, t = 1 .. T) \tag{9}$$

where

a_{ji} is the "gozinto-factor", i.e. the quantity of item j that is needed to produce one item i;

B is a large number greater than $\frac{\max \{ C_t \mid t = 1 .. T \}}{\min \{ p_j \mid j = 1 .. J \}}$,

C_t is the capacity of the machine in period t;

d_{jt} is the (external) demand for item j in period t;

h_j are the costs for holding one item j one period in inventory;

I_{jt} is the quantity of item j held in inventory at the end of period t (I_{j0} is the initial inventory);

J is the number of items;

p_j is the amount of capacity consumed by producing one item j;

q_{jt} is the quantity of item j to be produced in period t;

s_j are the setup costs for item j;

S(j) is the set of successors of item j, i.e. the set of items i where $a_{ji} > 0$;

T is the number of periods;

v_j is the (integral) lead time of item j ($v_j \geq 0$);

x_{jt} is a (binary) variable indicating whether a setup for item j occurs in period t ($x_{jt} = 1$) or not ($x_{jt} = 0$);

y_{jt} is a binary variable indicating whether the machine is setup for item j at the end of period t ($y_{jt} = 1$) or not ($y_{jt} = 0$) (y_{j0} is the initial setup state).

The objective of this model as it is expressed in (1) is to minimize the sum of setup and holding costs. Equations (2) are the inventory balances: At the end of a period we have in inventory what we had the period before, increased by what is produced in this period and decreased by what is consumed by internal and external demands in this period. In combination with the restrictions (9) which define inventory to be non-negative shortages are disallowed. To make sure that all internal demands can be satisfied promptly, the restrictions (3) have to be fulfilled: The lower bound of what is to be held in inventory at the end of a certain period is the sum of internal demands of the next v_j periods. Note, as long as we consider single-machine problems only (as we do here), positive lead times are not a must to guarantee an in time production. However, in the case of positive lead times we could use a more concise formulation to replace (2) and (3):

$$I_{j0} \geq \sum_{\tau=1}^{v_j} \sum_{i \in S(j)} (a_{ji} q_{i\tau}) \qquad (j = 1 .. J) \qquad (10)$$

$$I_{j1} = I_{j0} + q_{j1} - d_{j1} - \sum_{\tau=1}^{1+v_j} \sum_{i \in S(j)} (a_{ji} q_{i\tau}) \qquad (j = 1 .. J) \qquad (11)$$

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt} - \sum_{i \in S(j)} (a_{ji} q_{i(t+v_j)}) \qquad (j = 1 .. J, t = 2 .. T-v_j) \qquad (12)$$

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt} \qquad (j = 1 .. J,\ t = \max\{1, T-v_j+1\} .. T) \qquad (13)$$

The idea of this set of restrictions is to look in each period v_j periods ahead and to put aside from the inventory the internal demand of that particular future period. Although this point of view renders it possible to decrease the number of constraints when compared to (2) and (3), the (in)equalities (10) to (13) lead to a "wrong" assessment of holding costs because items that are being produced to meet an internal demand do not cause holding costs for being stored during the lead time. Note, that the set of feasible solutions is the same in both cases and that any optimal solution for one of the programs is an optimal solution for both programs. In the following we will stay with (2) and (3). In contrast to other authors, e.g. [Salomon et al. 1993, Tempelmeier 1992], we use (2) and (3) instead of

$$I_{jt} = I_{j(t-1)} + q_{j(t-v_j)} - d_{jt} - \sum_{i \in S(j)} (a_{ji}\, q_{it}).$$

The following drawbacks of these equations persuade us to replace them: External demands are interpreted as demands at the very end of periods. Hence, lead times need not to be considered to meet external demands. If lead times are considered to meet external demands, nevertheless, no external demand that cannot be satisfied by initial inventory can be permitted within the first v_j periods. Moreover, the last v_j periods cannot be used for production, because this production would mean to meet a demand outside the time horizon T.

The restrictions (4) ensure that the setup state of the machine is uniquely defined at the end of a period. Having in mind that we allow at most one changeover per period the constraints (5) guarantee that the machine is setup for an item whenever an item is produced. Noteworthy to say, that the setup state of the machine can be maintained during idle periods to avoid unnecessary setups and hence to avoid setup costs. Setup costs are to be charged whenever the setup state switches from zero to one. In combination with the minimize objective (1) the conditions (6) thus raise the setup flag of an item to one in such cases. Scarce capacities are respected by the inequalities (7). Decision variables are to be non-negative (or binary in the case of setup states) which is expressed in (8) and (9). In the sequel we assume the initial inventory and the initial setup state to be given.

3 A Randomized Regret Based Heuristic

In this section we will present a heuristic procedure to solve both multi-level, single-machine problems with and without initial inventory. The basic idea of our algorithm is to move backwards from period T to period one while accumulating the demands for each item separately. Within each period we choose one item with a positive cumulated demand to be produced and try to satisfy the complete demand. If no such item exists we simply step one period backwards. In the case that the whole cumulated demand for this item can be produced, i.e. the machine does not run out of capacity, we choose a second item for production. If two items are scheduled we step backwards one period to start all over again. If the machine capacity is exhausted we also step backwards one period to complete the production process of the selected item until the cumulated demand for this item is met. This scheme is repeatedly applied until period one is reached. Moving backwards instead of moving forwards from period one to period T is an appropriate approach for multi-level systems because

scheduling an item in period t causes internal demands for predecessing items (with positive lead times) in earlier periods. Once that an item is scheduled, the demand matrix may thus simply be updated by internal demands of directly predecessing items.

The major question to be answered is: "How to select an item for production out of a set of items with positive cumulated demand?". To decide which item to choose we make use of a regret measure [Drexl 1991] that estimates the item specific regret not to schedule the item in the current period. After computing the regrets for all items we randomly select an item in such a way that the higher the regret the more probable it is chosen. Despite the existence of items with a positive cumulated demand it ought happen that no item is chosen at all.

The regret measure that we use reflects several criteria which (in our opinion) determine the regret not to schedule an item in a certain period. These criteria are:

(a) Not to produce the cumulated (future) demand causes additional holding costs.

(b) Not to produce an item the machine is setup for (maybe) causes setup costs in other periods.

(c) Not to produce an item with positive depth, i.e. with predecessing items producible under the restriction of lead times, (maybe) leads to infeasibility due to the finite time horizon. This criterion may be neglected if the depth of an item is much smaller than the distance to the first period.

(d) Not to produce the cumulated demand, i.e. not making use of the machine capacity in the current period, (maybe) leads to infeasibility due to scarce capacity. It may not be paid attention to this criterion if the amount of capacity that is needed to produce an item (and all its predecessing items) is much smaller than the available capacity.

More formally the regret measure for item j in period t can be defined as follows: Let CD_{jt} denote the cumulated demand of item j not being satisfied, i.e. $CD_{jt} = \sum_{\tau=t}^{T} d_{j\tau} - \sum_{\tau=t+1}^{T} q_{j\tau}$. Let y denote the item the machine is setup for at the beginning of period t+1 (assume y = 0, i.e. the machine is setup for no item at all, when determining the regrets in period T). The depth of an item j (denoted as dep_j) can be measured by the maximal sum of lead times of the items along a path to item j, where a path to item j (denoted as P_j) is defined as a sequence of items $i_0 \rightarrow i_1 \rightarrow \ldots \rightarrow i_k \rightarrow \ldots \rightarrow i_K$ with $i_K = j$, $a_{i_{(k-1)}i_k} \geq 1$ for all $k \in \{1,\ldots,K\}$ and $a_{hi_0} = 0$ for all items $h \in \{1,\ldots,J\}$. This is to say, that the depth of an item j can be computed as a longest path to item j, i.e. $dep_j = \max \{ \sum_{k=0}^{K-1} v_{i_k} \mid i_0 \rightarrow i_1 \rightarrow \ldots \rightarrow i_k \rightarrow \ldots \rightarrow i_K$ is a path to item j$\}$. Let id_{ji} denote the internal demand for items i to be satisfied for producing one item j, i.e. $id_{jj} = 1$, $id_{ji} = \sum_{h \in S(i)} (a_{ih}\, id_{jh})$ if $i \neq j$ and there exists a path P_j such that $i \in P_j$ (where $i_k \in P_j$ is used to denote that item i_k is on path P_j), and $id_{ji} = 0$ if there is no such path P_j. The capacity that is consumed by producing one item j (and all its predecessing components) can now be defined as $cap_j = \sum_{i=1}^{J} (p_i\, id_{ji})$. The total amount of available capacity in period t (denoted as AC_t) can easily be computed as $AC_t = \sum_{\tau=1}^{t} C_\tau$. The regret measure for an item j in period t (denoted as r_{jt}) can in correspondence to the above criteria now be formulated as follows:

Case 1: $CD_{jt} > 0$ and $j \neq y$.

$$r_{jt} = \gamma_1 \frac{h_j CD_{jt}}{\max \{ s_i \mid \text{all items } i \}} \quad \text{(a)}$$

$$- \gamma_2 \frac{s_j}{\max \{ s_i \mid \text{all items } i \}} \quad \text{(b)}$$

$$+ \gamma_3 \frac{dep_j}{t - dep_j} \quad \text{(c)}$$

$$+ \gamma_4 \frac{CD_{jt}\, cap_j}{AC_t} \quad \text{(d)}$$

Case 2: $CD_{jt} > 0$ and $j = y$.

Just drop (b) in case 1.

Case 3: $CD_{jt} = 0$.

$r_{jt} = -\infty$

The real-valued parameters $\gamma_1,\ldots,\gamma_4$ control the influence of each of the criteria (a) to (d) where without loss of generality $0 \leq \gamma_1,\ldots,\gamma_4 \leq 1$ and $\sum_{i=1}^{4} \gamma_i = 1$ holds. According to what was said before, the focus of (a) and (b) is on the expenses that are incurred (or saved when setups are avoided) if a certain item is not scheduled, whereas (c) and (d) make feasible production plans more probable no matter how costly they are. Roughly speaking, the combination of these criteria leads to cheap and feasible production plans. Noteworthy, that only CD_{jt} needs to be computed over and over again while moving stepwise from period T to period one. The other components of the regret measure are known in advance and can thus be computed a priori. It should be mentioned that this definition is not a precise one. Instead of using $h_j\, CD_{jt}$ as an estimate of additional holding costs in (a) one could use $h_j \min \{ CD_{jt}, \lfloor \frac{C_t}{p_j} \rfloor \}$ which would be more precise because even if the item j would be produced in period t the capacity must not be exceeded and holding costs may therefore occur in any case. Computational studies turned out that a more precise estimate of additional holding costs does not improve the quality of the algorithm. Another "incorrect" point is that we consider the setup state at the beginning of the succeeding period t+1 although the setup state at the end of the preceding period t-1 would be of interest. This is an inherent problem of backward oriented procedures and seems to be unavoidable. Unless we consider sequence dependent setup costs (or times) the impact of this problem is negligible. Other (reasonable) definitions of the regret measure, e.g. using $\gamma_1 \frac{h_j CD_{jt}}{\max \{ h_i CD_{it} \mid \text{all items } i \}}$ and $\gamma_2 \frac{s_j}{\max \{ h_i CD_{it} \mid \text{all items } i \}}$ instead of (a) and (b), respectively, were tried out but gained no improvement.

Based on the above regret measure we now introduce a modified regret which allows us to take differences between the item specific regrets into account and hence enables a more sensitive control:

$$\rho_{jt} = \begin{cases} 0 & \text{if } r_{jt} = -\infty \\ (r_{jt} - \min \{ r_{it} \mid \text{items } i \text{ with } CD_{it} > 0 \} + \varepsilon)^{\delta} & \text{otherwise} \end{cases}$$

The modified regret measure is an item specific non-negative value and will be used for selecting an item with a positive cumulated demand for production. The parameter ε is a small positive value (e.g. $0.0001 \leq \varepsilon \leq 0.1$) which ensures that every item with a positive cumulated demand has a positive regret. This value may bias the regret if ε is of the same or a larger order of magnitude as the values $r_{jt} - \min \{ r_{it} \mid \text{items } i \text{ with } CD_{it} > 0 \}$, i.e. differences of the (unmodified) regret are smoothed if ε is too large. The real-valued parameter δ amplifies the

differences of the item specific (unmodified) regrets if $\delta \geq 1$ is chosen. In the case of $0 \leq \delta < 1$ the differences of the (unmodified) regrets are smoothed. A negative value of δ is unrealistic because this would mean that the higher the (unmodified) regret of an item the lower the modified regret and vice versa. A decision criterion based on such a regret measure would prefer to choose items with a low (unmodified) regret which would be contrary to the idea of regrets.

Let us now turn to describe the details of a backward oriented heuristic that is based on randomized regrets. As we will see later on, there are some strings attached when initial inventory is considered. For the sake of simplicity we will therefore start without considering initial inventory, i.e. initial inventory is assumed to be zero. After we will have developed a solution method for this simplified problem we will return to address the initial inventory problem. The main idea of the heuristic is to repeat the backward oriented scheduling algorithm over and over again to generate different production plans and to choose the best (cheapest) one. While moving backwards from period T to period one we should terminate the process of scheduling and start a new repetition as soon as we encounter a situation in which it is clear that no feasible production plan can be generated. Detecting such situations as soon as possible decidedly improves the performance. In our heuristic we test two infeasibility conditions: A situation cannot be completed to a feasible production plan if either the depth of an item with a positive cumulated demand is greater than or equal to the number of the current period, i.e. $dep_j \geq t$, or the sum of capacity needs is greater than the available capacity, i.e. $\sum_{j=1}^{J} (CD_{jt}\, cap_j) > AC_t$. A feasible production plan is eventually found if no item with a cumulated positive demand is left after period one is considered. The details of an implementation are given in the appendix.

The described procedure can be used to solve problem instances without initial inventory but fails if initial inventory should be considered. This would not reduce the class of feasible problems if we face single-level problems only because in the single-level case initial inventory can be used to meet the (external) demand in a simple way: The basic rule is to satisfy a demand in an early period before a demand in a late period is satisfied. A single-level problem instance with initial inventory can thus be converted into a problem instance without initial inventory by using a preprocessor that moves forward from period one to period T and "deletes" the entries in the item specific demand vectors as long as the demand can be met by the initial inventory. A scheduling algorithm may then take over to solve the resulting problem (where no initial inventory exists and some entries in the original demand matrix are decreased). Every feasible solution of the latter problem is (obviously) a feasible solution of the former one and vice versa. This is to say that if no feasible solution of the latter problem exists, the former one is insolvable, too. Moreover, the optimal objective function values of both problems differ by a constant (i.e. the holding costs for the initial inventory) only. For these reasons, both problems are equivalent.

Mapping initial inventory to demands in the multi-level case bears the risk of resulting in an infeasible problem although the original problem was solvable. The following simple example points this characteristic feature out: Suppose $J = 3$, $T = 4$, $C_1 = C_2 = 10$, $C_3 = C_4 = 1$, $v_1 = v_2 = v_3 = 1$, and $p_1 = p_2 = p_3 = 1$. Assume a linear structure with $(a_{ij})_{i,j = 1..J} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ and an initial inventory of $I_{10} = I_{20} = 0$ and $I_{30} = 3$. The (external) demand is given as $(d_{jt})_{\substack{j=1..J \\ t=1..T}} = \begin{pmatrix} 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$. Let the machine initially be setup for producing item

3. The matrix $(q_{jt})_{\substack{j=1..J \\ t=1..T}} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 2 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 \end{pmatrix}$ then defines a feasible production plan that meets the demands promptly. Having a closer look at this example turns out that in contrast to the single-level case a multi-level problem with initial inventory cannot be reduced to a problem without initial inventory during a preprocessing stage. The reason for this is that we have both external and internal demands for items, but only the external demands are known by time. To guarantee that a problem with initial inventory when turned into a problem without initial inventory is insolvable only in those cases where the original problem is not solvable, we may use the initial inventory to meet a demand (for an item) in a certain period if and only if the demands (for that item) in earlier periods can be satisfied by the initial inventory as well. Hence, initial inventory in multi-level problems cannot be reduced a priori. But even if we would (for the sake of simplicity) accept this property and would try to map the initial inventory to external demands (only), the problem of considering initial inventory during the runtime of the scheduling algorithm would not be circumvented because whenever the initial inventory is greater than the sum of external demands (see for instance the above example) we would face the problem of meeting (internal) demands by the initial inventory again. So, if generality should not be lost, i.e. problem instances with initial inventory are to be solved (which is the case in most real-world applications), a solution procedure for multi-level problems has initial inventory to take into account.

Let us now investigate if and how the above randomized regret based heuristic can be changed to solve problem instances with initial inventory to suboptimality. Reminding the idea of how initial inventory is taken into account in single-level problems, i.e. reminding that demands in early periods are met before demands in late periods are satisfied, the main idea of solving multi-level problems with initial inventory is as follows: Assume that for each item j the total number of items to be produced (denoted as tbp_j), i.e. the number of items that are demanded but not met by the initial inventory, is known. Whenever items j are scheduled for production in a period t the value tbp_j is to be decreased by q_{jt}, i.e. tbp_j is to be updated by $tbp_j - q_{jt}$. The value tbp_j can be seen as an upper bound of the cumulated demand CD_{jt} to be satisfied which was so far used to indicate the number of items to be produced until period t. Within each step from period to period CD_{jt} must therefore be computed as $CD_{jt} = \min \{ CD_{j(t+1)} + dem_{jt} , tbp_j \}$ where dem_{jt} is the sum of external and internal demand for item j in period t (see the algorithm given in the appendix). The backward oriented procedure thus pays off again because the (internal and external) demands of an item are now met by the initial inventory in those periods only in which the demands in earlier periods are also met.

The computation of tbp_j is not straightforward, i.e. $tbp_j = \sum_{t=1}^{T} \sum_{i=1}^{J} (id_{ij} d_{it}) - I_{j0}$ is incorrect. If the demand for an item j can be met by the initial inventory, we have to take into account that no internal demand for predecessing items $i \in P_j$ is caused. To compute tbp_j (j = 1 .. J) we therefore use the following simple program:

```
tbp_j := 0                          ( j = 1 .. J )    /* initialize all tbp_j */
inv_j := I_j0                       ( j = 1 .. J )    /* copy the initial inventory */
for t = 1 to T                                        /* move from period one to period T */
{   for all i ∈ {1,...,J}                             /* check every item */
    {   if ( d_it > 0 ) then                          /* external (and internal) demand is caused */
```

```
            add_tbp ( i , d_it )                    /* update tbp_j ( j = 1 .. J ) */
} }
where
subroutine add_tbp ( j , d )                        /* consider demand d for item j */
{  if ( d > inv_j ) then                            /* the demand exceeds the initial inventory */
   {  q := d - inv_j                                /* q items are to be produced */
      tbp_j := tbp_j + q                            /* update tbp_j */
      inv_j := 0                                    /* the initial inventory is exhausted */
      for all "direct predecessors k of j"          /* try to satisfy the internal demand */
         add_tbp ( k , a_kj q )                     /* move recursively along all paths to item j */
   } else inv_j := inv_j - d                        /* demand can be met by the initial inventory */
}
```

Noteworthy to say, that this program can run before the scheduling algorithm starts, i.e. tbp_j (j = 1 .. J) can be computed in advance. If a positive value of inv_j remains after the above routine was executed, no production of items j must take place, i.e. all demands for item j are met by the initial inventory. The remaining inv_j items j are to be held in inventory for all T periods of time (I_{jT} may thus be a positive value) and therefore cause holding costs.

The major impact of concerning initial inventory is upon the infeasibility checks that are made to detect situations in which no feasible production plan can be reached any more. Neither the comparison of the depths of the items with the number of the current period nor the test for insufficient capacity is a valid infeasibility condition when initial inventory is considered. The problem in both cases is that some of the internal demands that are required to be satisfied in order to produce a particular item may be met by the initial inventory. Hence, the depth of an item as well as its capacity needs may be less than predicted. As it was pointed out earlier, though we can compute the precise number of items to be produced, we cannot compute the precise time that the production takes place (otherwise we would have an exact solution method and not a heuristic). Both infeasibility checks must therefore be omitted. This is to say, that due to the criterion (c) the (unmodified) regret of an item j whose depth exceeds the time horizon and t - dep_j thus evaluates to a negative value (or is set to one if t = dep_j) tends to be small. A similar problem is faced when an item is scheduled for production and the lead time of a directly predecessing item exceeds the time horizon (when the demand matrix is to be updated). Again, it cannot be decided whether or not this leads to infeasibility. Although it is apparent that such predecessing items must be taken from the initial inventory it is unclear if all of these items can indeed be taken from the initial inventory until period one is considered. Thus again the scheduling algorithm cannot be terminated to start a new repetition. The demand matrix is to be updated only in those cases where the lead time does not exceed the time horizon. If period one is eventually considered, a final infeasibility check can be performed by testing the variables tbp_j. If there exists at least one positive value, the production plan is infeasible (because more items should have been produced) and can thus be dismissed.

Finally, the evaluation of a production plan, i.e. the computation of the objective function value, shall now be discussed. While the setup costs can trivially be computed by scanning a production plan from period

one to period T, some efforts have to be made for the determination of the holding costs. In the case of single-level problems (without initial inventory) the holding costs can be calculated by

$$\sum_{t=1}^{T} \sum_{j=1}^{J} (h_j (q_{jt} - d_{jt}) (T + 1 - t)) .$$

If multi-level problems are considered, the subject becomes a bit more complex because items that are produced to meet an internal demand may have a positive lead time, i.e. the items of the kind j are to be held in inventory for at least v_j periods. Thus

$$\sum_{t=1}^{T} (\sum_{j=1}^{J} (h_j (q_{jt} - d_{jt}) - \sum_{i=1}^{J} (h_i a_{ij} q_{jt})) (T + 1 - t)) + \sum_{j=1}^{J} (h_j I_{j0} T)$$

defines the holding costs in the multi-level case.

4 Computational Study

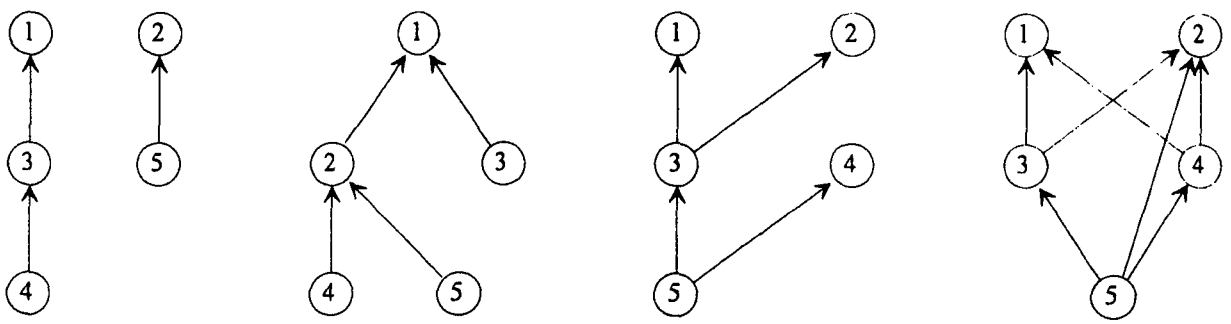
To study the presented heuristic it was coded in C [Kernighan and Ritchie 1990] and executed on a 486 PC with 25 MHz. The computational results were promising:

We tested a total of 144 sample problems with 5 items and 10 periods of time and without initial inventory. Since the only difference between the heuristic which solves problems with initial inventory and the one which solves problems without initial inventory is that the former one computes the number of items to be produced in a preprocessing stage and performs no infeasibility checks, computational results for problem instances with initial inventory are not given. The control parameters ε and δ of the heuristic were chosen at random from the intervals [0.0001 , 0.1] and [0 , 10], respectively. 4 different product structures were considered (see figure below). For each of these structures we defined 3 problems with external demand for end items only and 3 problems with external demand for all items, i.e. a total of 24 combinations were constructed. The demand patterns were chosen as follows:

(1) External demand in period 10.

(2) External demand in periods 6 and 10.

(3) External demand in periods 6, 8 and 10.



Linear Structure Assembly Structure Divergent Structure General Structure

Table 4.1: Setup and holding costs

item	setup costs	holding costs
1	30	5
2	20	4
3	20	3
4	10	2
5	10	1

A problem is named by a triple of the form $\sigma/\pi/\nu$ where $\sigma \in \{ L, A, D, G \}$ indicates the underlying product structure (i.e. linear, assembly, divergent or general), $\pi \in \{ E, A \}$ indicates if external demand occurs for end items only or for all items, and $\nu \in \{ 1, 2, 3 \}$ indicates the demand pattern in correspondence to what is defined above. Each of these 24 problems were tested in combination with 6 different sets of data:

(a) Holding and setup costs are defined as in table 4.1. Lead times and production coefficients are equal to one in all cases, i.e. $v_j = 1$ and $a_{ji} = 1$. The production of one item (of any kind) consumes one capacity unit, i.e. $p_j = 1$ for all items, while the capacity of the machine is assumed to be constant over time. More precisely, we assume the capacity constraints per time period to be defined as in table 4.2. The external demands per period (with respect to the demand pattern ν) are assumed to be 20 per item in the case that $\pi = E$ and 10 per item in the case that $\pi = A$.

(b) The same data set as (a) out of setup costs being multiplied by 20.

(c) The same data set as (a) out of external demand sizes being half of what is defined above, i.e. 10 items if $\pi = E$ and 5 items if $\pi = A$.

(d) The same data set as (b) out of external demand sizes being chosen as in (c).

(e) The same data set as (c) out of $p_j = 0.5$ for all items and all production coefficients are doubled, i.e. $a_{ji} = 2$ for all items j and i with respect to the product structures defined above.

(f) The same data set as (d) out of p_j and a_{ji} being chosen alike (e).

Each of these 144 problems has more than one feasible solution.

Table 4.2: Capacity constraints per period of time

	$\pi = E$			$\pi = A$		
	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 1$	$\nu = 2$	$\nu = 3$
$\sigma = L$	35	35	100	35	35	100
$\sigma = A$	35	35	100	35	35	100
$\sigma = D$	35	100	100	35	100	100
$\sigma = G$	35	100	200	100	100	200

Tables 4.3 to 4.8 provide the computational results when 1000 repetitions are performed. The objective function values are compared to the optimal solutions which were computed by means of standard MIP-solvers (i.e. OSL [IBM 1992] and LINDO [Schrage 1991]). As a measure of the quality of a solution we use

$$\text{deviation} = 100 * \frac{F^*_H - F^*_{opt}}{F^*_{opt}}$$

where F^*_H denotes the best solution found by the heuristic and F^*_{opt} denotes the optimal solution computed by the standard solvers. Furthermore we give the infeasibility ratio, i.e. the percentage of iterations which did not find a feasible production plan. The utilization of the machine capacity is determined by

$$\text{scarcity} = 100 * \sum_{t=1}^{T} \sum_{j=1}^{J} \frac{(d_{jt}\, cap_j)}{AC_T}$$

though a low value does not mean that a feasible production plan can easily be found. Even if this value is very low it may happen that no production plan can be found at all. The reason for this is that as long as shortages are not allowed positive lead times as well as demands in early periods force capacity units (in late periods) to be unused.

Table 4.3: Deviation for problem instances with v = 1 and data sets (a) to (f)

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/1	28.30	3.80	0	0	0	0	5.35
L/A/1	10.34	1.50	7.89	0.79	5.88	0.76	4.53
A/E/1	15.38	2.46	0	0	0	0	2.97
A/A/1	28.57	6.07	6.82	0	4.48	0.73	7.79
D/E/1	28.41	6.74	8.33	1.03	10.26	0.46	9.21
D/A/1	25.40	2.93	14.63	1.57	10.91	0.50	9.32
G/E/1	24.65	9.50	0	0	13.87	2.43	8.41
G/A/1	8.89	1.85	7.41	1.01	12.79	0	5.33
average	21.24	4.36	5.64	0.55	7.27	0.61	6.61

Table 4.3 provides the results of problem instances with v = 1 when applied to different data sets. As it can be seen in table 4.4, only few iterations lead to an infeasible production plan. One could therefore say, that these problems were easily solved by the heuristic. Problem instances applied to data set (a) were badly solved when compared to the other results.

Table 4.4: Infeasibility ratio and scarcity for problem instances with v = 1 and data sets (a) to (f)

	(a)	(b)	(c)	(d)	(e)	(f)
L/E/1	0 / 28.57	0 / 28.57	0 / 14.29	0 / 14.29	0 / 28.57	0 / 28.57
L/A/1	0 / 25.71	0 / 25.71	0 / 12.86	0 / 12.86	0 / 21.43	0 / 21.43
A/E/1	0 / 28.57	0 / 28.57	0 / 14.29	0 / 14.29	0 / 37.14	0 / 37.14
A/A/1	0.20 / 31.43	0.40 / 31.43	0.30 / 15.71	0.40 / 15.71	0.40 / 30.00	1.00 / 30.00
D/E/1	0 / 45.71	0 / 45.71	0 / 22.86	0 / 22.86	0 / 48.57	0 / 48.57
D/A/1	0 / 31.43	0 / 31.43	0 / 15.71	0.10 / 15.71	0 / 30.00	0.10 / 30.00
G/E/1	0.70 / 62.86	8.50 / 62.86	0 / 31.43	0.50 / 31.43	0.20 / 80.00	1.00 / 80.00
G/A/1	2.60 / 16.00	17.60 / 16.00	5.20 / 8.00	20.60 / 8.00	7.30 / 17.50	19.90 / 17.50

It is interesting to note, that those G/A/1-instances which were applied to data sets (b), (d) and (f) lead to a high number of infeasible solutions but were solved nearly optimal, i.e. with a deviation of less than 2 %.

Table 4.5: Deviation for problem instances with $v = 2$ and data sets (a) to (f)

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/2	35.90	34.70	11.11	65.91	6.67	40.40	32.45
L/A/2	14.29	25.95	14.10	30.42	10.68	30.00	20.91
A/E/2	22.38	41.55	2.33	60.18	3.61	41.26	28.55
A/A/2	23.75	26.76	9.38	31.72	9.93	28.38	21.65
D/E/2	12.36	29.30	11.32	32.52	21.95	29.80	22.88
D/A/2	23.88	31.29	20.24	32.53	22.81	26.85	26.27
G/E/2	0	20.34	0	18.79	9.92	27.68	12.79
G/A/2	15.63	26.09	14.41	14.90	23.53	24.05	19.77
average	18.52	29.50	10.36	35.87	13.64	31.05	23.16

The results of problem instances with $v = 2$ are given in table 4.5. Most of these problems, especially those with external demand for all items, were hardly solved by the heuristic as indicated in table 4.6.

Table 4.6: Infeasibility ratio and scarcity for problem instances with $v = 2$ and data sets (a) to (f)

	(a)	(b)	(c)	(d)	(e)	(f)
L/E/2	0 / 57.14	0 / 57.14	0 / 28.57	0 / 28.57	0 / 57.14	0 / 57.14
L/A/2	16.80 / 51.43	33.40 / 51.43	15.80 / 25.71	29.70 / 25.71	15.20 / 42.86	29.00 / 42.86
A/E/2	0.30 / 57.14	0.10 / 57.14	0 / 28.57	0 / 28.57	0 / 74.29	0 / 74.29
A/A/2	40.50 / 62.86	64.60 / 62.86	32.30 / 31.43	34.30 / 31.43	28.60 / 60.00	34.70 / 60.00
D/E/2	23.40 / 32.00	30.30 / 32.00	23.20 / 16.00	34.10 / 16.00	87.80 / 34.00	40.50 / 34.00
D/A/2	79.70 / 22.00	60.30 / 22.00	74.80 / 11.00	61.30 / 11.00	90.70 / 21.00	61.70 / 21.00
G/E/2	18.10 / 44.00	38.80 / 44.00	19.70 / 22.00	30.10 / 22.00	85.00 / 56.00	51.60 / 56.00
G/A/2	73.00 / 32.00	58.10 / 32.00	60.10 / 16.00	42.80 / 16.00	80.50 / 35.00	50.80 / 35.00

Table 4.7 presents the results of problem instances with $v = 3$ when applied to different data sets. As one can see in table 4.8, it was hard to solve these problems by the heuristic.

Table 4.7: Deviation for problem instances with $v = 3$ and data sets (a) to (f)

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/3	13.40	28.67	14.29	38.75	5.62	34.74	22.58
L/A/3	17.78	12.18	18.10	15.85	13.48	14.47	15.31
A/E/3	11.93	39.74	9.52	49.79	4.69	46.05	26.95
A/A/3	15.25	23.12	6.16	14.08	11.17	18.07	14.64
D/E/3	16.56	11.56	13.10	19.44	22.40	14.56	16.27
D/A/3	30.10	6.18	24.17	14.18	22.29	15.29	18.70
G/E/3	3.17	19.27	6.03	28.71	10.24	19.76	14.53
G/A/3	12.24	19.37	12.27	20.97	19.62	20.64	17.52
average	15.05	20.01	12.96	25.22	13.69	22.95	18.31

The problem instances that we used here, especially those with $v = 2$ and $v = 3$, could be termed hard because many items had to be scheduled within a narrow time horizon. It is thus a promising thing to note that the heuristic was able to find a feasible solution in all cases. Easier problems which are expected to be solved

better when compared to the results above were not tested here because of two reasons: First, we could have decreased the number of items, but considering less than 5 items to study multi-level problems is not an interesting thing to do. Second, we could have increased the number of time periods (and the number of items), but larger problems could not have been solved to optimality within reasonable time.

Table 4.8: Infeasibility ratio and scarcity for problem instances with v = 3 and data sets (a) to (f)

	(a)	(b)	(c)	(d)	(e)	(f)
L/E/3	41.80 / 30.00	21.50 / 30.00	32.90 / 15.00	19.20 / 15.00	22.10 / 30.00	24.20 / 30.00
L/A/3	59.80 / 27.00	64.70 / 27.00	60.50 / 13.50	66.30 / 13.50	49.50 / 22.50	66.80 / 22.50
A/E/3	25.30 / 30.00	15.10 / 30.00	26.50 / 15.00	13.40 / 15.00	27.40 / 39.00	12.60 / 39.00
A/A/3	42.00 / 33.00	37.90 / 33.00	47.80 / 16.50	39.90 / 16.50	39.10 / 31.50	39.70 / 31.50
D/E/3	63.30 / 48.00	54.10 / 48.00	43.00 / 24.00	15.80 / 24.00	83.10 / 51.00	36.90 / 51.00
D/A/3	72.60 / 33.00	31.80 / 33.00	55.90 / 16.50	19.10 / 16.50	75.20 / 31.50	23.70 / 31.50
G/E/3	93.70 / 33.00	94.80 / 33.00	84.20 / 16.50	60.80 / 16.50	97.90 / 42.00	92.30 / 42.00
G/A/3	92.40 / 24.00	97.00 / 24.00	88.50 / 12.00	74.00 / 12.00	96.30 / 26.25	79.70 / 26.25

Noteworthy, that the time needed by the heuristic to solve the above problems was between 4 to 8 seconds per problem running on a 486 PC with 25 MHz. On the other hand standard MIP-solvers (i.e. OSL [IBM 1992] and LINDO [Schrage 1991]) took several days to solve all the sample problems running on an IBM RS/6000-550 AIX-workstation. In some cases our computer run out of memory space when we used OSL while in some other cases LINDO terminated with wrong results because of errors introduced by rounding (we assume). So the advantage of using a heuristic is not only the short response time, it is the capability of solving problems that cannot be solved by exact methods as well.

5 Conclusion and Future Work

On the basis of the PLSP we have presented a mixed-integer model for multi-level, single-machine lot sizing and scheduling problems. To solve problem instances to suboptimality an efficient backward oriented randomized regret based heuristic was developed. Surprisingly, (almost) all components that have an influence on the regret measure could have been determined in advance, so that the calculation of regrets does not decrease the performance when compared to a corresponding single-level heuristic. The backward oriented solution approach paid off because the multi-level nature of the problem instances has neatly been taken into account by simply updating the demand matrix. Another advantageous side-effect of the backward oriented algorithm is that it facilitates the early detection of infeasible production plans. It was shown that in contrast to single-level problems the consideration of initial inventory is a crucial aspect since neglecting this case reduces the class of solvable problems. Again, the backward oriented method, in combination with the capability to compute the precise number of items to be produced a priori, helps to generate feasible production plans. Unfortunately, it turned out that solving problems with initial inventory allows no infeasibility checks to be done until a complete production plan is scheduled.

Future work should be about the extension of the presented model and heuristic for meeting real-world requirements like setup times, sequence dependent setup costs and times, capacitated inventory or shortage management. Moreover, the heuristic should be applied to multi-machine problems as well. The random choice of the regret parameters could be guided by selection criteria [Haase 1993] to improve the results. Work in this field would also be of interest.

Acknowledgement

This work was heavily inspired by Knut Haase's thesis work about lot sizing and scheduling for production planning where he developed a randomized regret based heuristic for single-level PLSP-instances. The entire manuscript was read by Prof. Dr. Andreas Drexl who gave very helpful comments.

Appendix: Implementation of the Heuristic

/* Randomized Regret Based Heuristic for Multi-Level, Single-Machine Problems without Initial Inventory */
/* for the sake of simplicity we assume positive lead times for all items */
/* *Initialization:* */
$F^* := \infty$ /* F^* denotes the best objective function value */
repeat #repetitions /* repeat the scheduling algorithm */
{ /* *Initialization:* */
$CD_{j(T+1)} := 0$ ($j = 1 .. J$) /* there is no demand when we start */
$dem_{jt} := d_{jt}$ ($j = 1 .. J, t = 1 .. T$) /* copy demand matrix */
$q_{jt} := 0$ ($j = 1 .. J, t = 1 .. T$) /* a new production plan */
$t := T$ /* start in period T */
$y := 0$ /* the machine is setup for no item */
setups := 0 /* number of setups within the current period */
carry := false /* carry on the production of an item */
"choose $\gamma_1,...,\gamma_4,\varepsilon$ and δ at random" /* select parameters from specified intervals */
/* *Move backwards from period T to period one:* */
while ($t \geq 1$) **do** /* stop after period one is considered */
{ $CD_{jt} := CD_{j(t+1)} + dem_{jt}$ ($j = 1 .. J$) /* accumulate the unsatisfied demand */
if ($\exists j : (dep_j \geq t)$) **then goto** "next repetition" /* infeasibility check */
if ($\sum_{j=1}^{J} (CD_{jt}\, cap_j) > AC_t$) **then goto** "next repetition" /* infeasibility check */
"compute ρ_{jt}" ($j = 1 .. J$) /* determine the item specific regrets */
if (not carry) **then** /* select a new (first) item i */
"select an item i at random proportional to ρ_{jt}" /* maybe no item is selected */
else "select item i = y" /* continue the production of an item i */

```
if ( "an item i is selected") then
{  if ( ( p_i CD_it ) ≤ C_t ) then                    /* the total demand can be produced */
   {  setups := setups + 1                            /* a setup must take place */
      schedule ( i , CD_it )                          /* schedule item i */
      if ( ∃ j ≠ i : ( CD_jt > 0 ) ) then             /* try to produce a second item */
      {  if ( setups < 2 ) then                       /* at most two items are allowed */
         {  if ( t = 1 ) then                         /* take the initial setup state into account */
            {  if ( "the machine is initially setup for an item h") then
               {  if ( CD_ht > 0 ) then "select the item h"   /* there is a positive demand for item h */
               }
               else                                   /* the machine is not setup */
               {  if ( setups = 0 ) then              /* at most one changeover */
                     "select an item h ≠ i at random proportional to ρ_jt"
            } }
            else                                      /* t > 1 */
            {  "select an item h ≠ i at random proportional to ρ_jt"
               setups := setups + 1                   /* a setup must take place */
            }
            if ("an item h is selected") then
            {  if ( ( p_h CD_ht ) ≤ C_t - ( p_i q_it ) ) then   /* the total demand can be produced */
                  schedule ( h , CD_ht )              /* schedule item h */
               else                                   /* the capacity is exceeded */
                  schedule ( h , ⌊ (C_t - ( p_i q_it )) / p_h ⌋ )   /* schedule item h */
      } } } }
      else                                            /* the capacity is exceeded */
      {  setups := setups + 1                         /* a setup must take place */
         schedule ( i , ⌊ C_t / p_i ⌋ )               /* schedule item i */
      }
      if ( setups < 2 ) then setups := 0              /* at most one changeover is done in t */
      else setups := 1                                /* one setup must be done in t-1 */
   }
   else setups := 0                                   /* t is an idle period */
   t := t - 1                                         /* step backwards */
}                                                     /* leave loop when t = 0 */
if ( ( setups = 1 ) and "the machine is not initially setup for y") then   /* infeasibility check */
   goto "next repetition"
if ( ∃ j : ( CD_j1 > 0 ) ) then goto "next repetition"   /* infeasibility check */
"evaluate the production plan"                        /* compute the objective function value F */
if ( F < F* ) then                                    /* a cheaper production plan is found */
```

```
    {   F* := F
        "memorize the current production plan"
}   }                                          /* end of repetition */
display "the best production plan"
where
subroutine schedule ( j , q )                  /* schedule q items j */
{   q_jt := q                                  /* produce q items in period t */
    if ( q < CD_jt ) then                      /* the total demand cannot be produced */
    {   CD_jt := CD_jt - q                     /* decrease the cumulated demand */
        carry := true                          /* continue the production in t-1 */
    }
    else                                       /* the total demand can be produced */
    {   CD_jt := 0                             /* decrease the cumulated demand */
        carry := false                         /* there is no production to be continued */
    }
    y := j                                     /* change the setup state */
    for all "direct predecessors k of j"       /* raise the internal demand */
        dem_k(t-v_k) := dem_k(t-v_k) + ( a_kj q )   /* update the demand matrix */
}
```

References

Bitran, G. R., Matsuo, H., (1986), The Multi-Item Capacitated Lot Size Problem: Error Bounds on Manne's Formulation, Management Science, Vol. 32, pp. 350-359

de Both, M. A., Gelders, L. F., van Wassenhove, L. N., (1984), Lot Sizing under Dynamic Demand Conditions: A Review, Engineering Costs and Production Economics, Vol. 8, pp. 165-187

Dinkelbach, W., (1964), Zum Problem der Produktionsplanung in Ein- und Mehrproduktunternehmen, Würzburg, 2. Auflage

Drexl, A., (1991), Scheduling of Project Networks by Job Assignment, Management Science, Vol. 37, pp. 1590-1602

Drexl, A., Haase, K., (1992), A New Type of Model for Multi-Item Capacitated Dynamic Lotsizing and Scheduling, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 286

Fleischmann, B., (1990), The Discrete Lot Sizing and Scheduling Problem, European Journal of Operational Research, Vol. 44, pp. 337-348

Günther, H.-O., (1987), Planning Lot Sizes and Capacity Requirements in a Single-Stage Production System, European Journal of Operational Research, Vol. 31, pp. 223-231

Haase, K., (1993), Lotsizing and Scheduling for Production Planning, Ph.D. thesis, University of Kiel

Haehling von Lanzenauer, C., (1970), A Production Scheduling Model by Bivalent Linear Programming, Management Science, Vol. 17, pp. 105-111

IBM Corporation, (1992), Optimization Subroutine Library (OSL), Guide and Reference, Release 2

Karmarkar, U. S., Kekre, S., Kekre, S., (1987), The Dynamic Lot Sizing Problem with Startup and Reservation Costs, Operations Research, Vol. 35, pp. 389-398

Kernighan, B. W., Ritchie, D. M., (1990), Programmieren in C, München, Hanser, 2. Auflage

de Matta, R., Guignard, M., (1989), Production Scheduling with Sequence Independent Changeover Cost, Working Paper, The Wharton School, University of Pennsylvania

Salomon, M., (1991), Deterministic Lot Sizing Models for Production Planning, Lecture Notes in Economics and Mathematical Systems, Vol. 355, Berlin, Springer

Salomon, M., Kuik, R., van Wassenhove, L. N., (1993), Statistical Search Methods for Lotsizing Problems, Annals of Operations Research, Vol. 41, pp. 453-468

Schrage, L., (1991), LINDO User's Manual for Linear, Integer and Quadratic Programming Release 5.0, Scientific Press

Tempelmeier, H., (1992), Material-Logistik, Berlin, Springer, 2. Auflage