Kimms, Alf; Drexl, Andreas

Some insights into proportional lot sizing and scheduling

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 406

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

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 406

Some Insights into Proportional Lot Sizing and Scheduling

A. Kimms and A. Drexl

No. 406

Some Insights into Proportional Lot Sizing and Scheduling

A. Kimms and A. Drexl

September 1996

Alf Kimms, Andreas Drexl
Lehrstuhl für Produktion und Logistik, Institut für Betriebswirtschaftslehre,
Christian–Albrechts–Universität zu Kiel, Olshausenstr. 40, 24118 Kiel, Germany
email: Kimms@bwl.uni–kiel.de
Drexl@bwl.uni–kiel.de
URL: http://www.wiso.uni–kiel.de/bwlinstitute/Prod
ftp://ftp.wiso.uni–kiel.de/pub/operations–research

Abstract

This paper deals with proportional lot sizing and scheduling (PLSP) and gives some insights into the properties of this problem. Such insights may be useful for developing heuristic and/or exact solution procedures. The emphasis of this paper is on the multi–level, multi–machine case. We provide a mixed–integer programming model, relate it to other models that can be found in the literature, and discuss characteristics which make solving instances of the PLSP–model become a hard task.

Keywords: Multi–level lot sizing, scheduling, initial inventory, lot splitting, valid constraints, postprocessing

1 Basic Assumptions

Several items are to be produced in order to meet some known (or estimated) dynamic demand without backlogs and stockouts. Precedence relations among these items define an acyclic gozinto–structure of the general type. In contrast to many authors who allow demand for end items only, now, demand may occur for all items including component parts. The finite planning horizon is subdivided into a number of discrete time periods. Positive lead times are given due to technological restrictions such as cooling or transportation for instance. Furthermore, items share common resources. Some (maybe all) of them are scarce. The capacities may vary over time. Producing one item requires an item–specific amount of the available capacity. All data are assumed to be deterministic.

Items which are produced in a period to meet some future demand must be stored in inventory and thus cause item–specific holding costs. Most authors assume that the holding costs for an item must be greater than or equal to the sum of the holding costs for all immediate predecessors. They argue that holding costs are mainly opportunity costs for capital which occurs no matter a component part is assembled or not. Two reasons persuade us to make no particular assumptions for holding costs. First, as it is usual in the chemical industry for instance, keeping some component parts in storage may require ongoing additional effort such as cooling, heating, or shaking. While these parts need no special treatment when processed, storing component parts might be more expensive than storing assembled items. Second, operations such as cutting tin mats for instance make parts smaller and often easier to handle. The remaining "waste" can often be sold as raw material for other manufacturing processes. Hence, opportunity costs may decrease when component parts are assembled. However, it should be made clear that the assumption of general holding costs is the most unrestrictive one. All models and methods developed under this assumption work for more restrictive cases as well.

Each item requires at least one resource for which a setup state has to be taken into account. Production can only take place if a proper state is set up. Setting a resource up for producing a particular item incurs item–specific

setup costs which are assumed to be sequence independent. Setup times are not considered. Once a certain setup action is performed, the setup state is kept up until another setup changes the current state. Hence, same items which are produced having some idle time in–between do not enforce more than one setup action. To get things straight, note that some authors use the word changeover instead of setup in this context.

The most fundamental assumption here is that for each resource at most one setup may occur within one period. Hence, at most two items sharing a common resource for which a setup state exists may be produced per period. Due to this assumption, the problem is known as the proportional lot sizing and scheduling problem (PLSP) [8, 14, 24]. By choosing the length of each time period appropriately small, the PLSP is a good approximation to a continuous time axis. It refines the well–known discrete lot sizing and scheduling problem (DLSP) [7, 10, 17, 26, 31] as well as the continuous setup lot sizing problem (CSLP) [3, 20, 19]. Both assume that at most one item may be produced per period. All three models could be classified as small bucket models since only a few (one or two) items are produced per period. In contrast to this, the well–known capacitated lot sizing problem (CLSP) [6, 9, 11, 16, 25, 27, 28, 33, 34] represents a large bucket model since many items can be produced per period. Remember, the CLSP does not include sequence decisions and is thus a much "easier" problem. An extension of the single–level CLSP with partial sequence decisions can be found in [13]. In [15] a large bucket single–level lot sizing and scheduling model is discussed.

2 A Mixed–Integer Programming Model

An important variant of the PLSP is the one with multiple machines (PLSP–MM) [24]. Several resources (machines) are available and each item is produced on an item–specific machine. This is to say that there is an unambiguous mapping from items to machines. Of course, some items may share a common machine. Special cases are the single–machine problem for which models and methods are given in [22, 23], and the problem with dedicated machines where items do not share a common machine. For the latter optimal solutions can be easily computed with a lot–for–lot policy [21].

Let us first introduce some notation. In Table 1 the decision variables are defined. Likewise, the parameters are explained in Table 2. Using this notation, we are now able to present a MIP–model formulation.

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

subject to

Symbol	Definition
I_{jt}	Inventory for item j at the end of period t.
q_{jt}	Production quantity for item j in period t.
x_{jt}	Binary variable which indicates whether a setup for item j occurs in period t ($x_{jt} = 1$) or not ($x_{jt} = 0$).
y_{jt}	Binary variable which indicates whether machine m_j is set up for item j at the end of period t ($y_{jt} = 1$) or not ($y_{jt} = 0$).

Table 1: Decision Variables for the PLSP–MM

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

$$I_{jt} \geq \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+1}^{\min\{t+v_j, T\}} a_{ji} q_{i\tau} \qquad \begin{matrix} j = 1, \ldots, J \\ t = 0, \ldots, T-1 \end{matrix} \tag{3}$$

$$\sum_{j \in \mathcal{J}_m} y_{jt} \leq 1 \qquad \begin{matrix} m = 1, \ldots, M \\ t = 1, \ldots, T \end{matrix} \tag{4}$$

$$x_{jt} \geq y_{jt} - y_{j(t-1)} \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{5}$$

$$p_j q_{jt} \leq C_{m_j t}(y_{j(t-1)} + y_{jt}) \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{6}$$

$$\sum_{j \in \mathcal{J}_m} p_j q_{jt} \leq C_{mt} \qquad \begin{matrix} m = 1, \ldots, M \\ t = 1, \ldots, T \end{matrix} \tag{7}$$

$$y_{jt} \in \{0, 1\} \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{8}$$

$$I_{jt}, q_{jt}, x_{jt} \geq 0 \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{9}$$

The objective (1) is to minimize the sum of setup and holding costs. Equations (2) are the inventory balances. At the end of a period t we have in inventory what was in there at the end of period $t-1$ plus what is produced minus external and internal demand. To fulfill internal demand we must respect positive lead times. Restrictions (3) guarantee so. Constraints (4) make sure that the setup state of each machine is uniquely defined at the end of each period. Those periods in which a setup happens are spotted by (5). Note that idle periods may occur in order to save setup costs. Due to (6) production can only take place if there is a proper setup state either at the beginning or at the end of a particular period. Hence, at most two items can be manufactured on each

Symbol	Definition
a_{ji}	"Gozinto"–factor. Its value is zero if item i is not an immediate successor of item j. Otherwise, it is the quantity of item j that is directly needed to produce one item i.
C_{mt}	Available capacity of machine m in period t.
d_{jt}	External demand for item j in period t.
h_j	Non–negative holding cost for having one unit of item j one period in inventory.
I_{j0}	Initial inventory for item j.
$\mathcal{J}_m$	Set of all items that share the machine m, i.e. $\mathcal{J}_m \stackrel{def}{=} \{j \in \{1,\ldots,J\} \mid m_j = m\}$.
J	Number of items.
M	Number of machines.
m_j	Machine on which item j is produced.
p_j	Capacity needs for producing one unit of item j.
s_j	Non–negative setup cost for item j.
$\mathcal{S}_j$	Set of immediate successors of item j, i.e. $\mathcal{S}_j \stackrel{def}{=} \{i \in \{1,\ldots,J\} \mid a_{ji} > 0\}$.
T	Number of periods.
v_j	Positive and integral lead time of item j.
y_{j0}	Unique initial setup state.

Table 2: Parameters for the PLSP–MM

machine per period. Capacity constraints are formulated in (7). Since the right hand side is a constant, overtime is not available. (8) define the binary–valued setup state variables, while (9) are simple non–negativity conditions. The reader may convince himself that due to (5) in combination with (1) setup variables x_{jt} are indeed zero–one valued. Hence, non–negativity conditions are sufficient for these. For letting inventory variables I_{jt} be non–negative backlogging cannot occur.

A DLSP–like model can be derived from the PLSP–model by adding

$$p_j q_{jt} = C_{m_j t} y_{jt} \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 1,\ldots,T \end{array} \tag{10}$$

to the set of constraints. It can easily be verified that now at most one item can be produced per period. Typical for the DLSP is the so–called "all–or–nothing" production. Having these equations introduced, the resulting model can be simplified. (6) and (7) are superfluous and can be dropped. Moreover, variables q_{jt} can be eliminated by substitution. Since adding constraints does

not decrease the optimal objective function value, we have given a proof that the optimal objective function value of a PLSP–instance is less than or equal to the optimal objective function value of the corresponding DLSP–instance. Moreover, if lots of the same item are produced in different periods, then idle periods cannot occur in–between without enforcing a new setup which is in contrast to the PLSP. Furthermore, whereas in the PLSP–model the unequals sign in (4) can be replaced with an equals sign without changing anything, the DLSP–model must have an unequals sign. It should be remarked, that traditional DLSP–model formulations do not consider time varying capacities, i.e. $C_{m1} = \ldots = C_{mt} = \ldots = C_{mT}$ for all $m = 1, \ldots, M$.

A CSLP–model formulation can be derived from the PLSP–model be adding

$$p_j q_{jt} \leq C_{m_j t} y_{jt} \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{11}$$

to the set of constraints. The fact that at most one item can be produced per period should again be clear when looking at (4). Similar to what was done when we derived the DLSP–model, some steps of simplification can reduce the size of the resulting model. As in the DLSP–case above, (6) and (7) can be dropped. Again, we now have a formal proof that the optimal objective function value of a PLSP–instance is always less than or equal to the objective function value of the corresponding CSLP–instance. In analogy to the PLSP, the CSLP also allows idle periods in order to save setup costs.

In summary, the PLSP can be seen as a generalization of the DLSP and the CSLP. With minor modifications all methods developed for the PLSP should be applicable to the latter ones as well. However, the reverse is not true. Efficient methods making use of restrictive assumptions need not work well for the PLSP. It comes out that an optimal solution for a PLSP instance is less than or equal to an optimal solution of the DLSP or the CSLP using the same data set. Since heuristic methods for the multi–level DLSP or the multi–level CSLP do not exist, there is no such benchmark for suboptimal procedures available for the PLSP.

To transform the PLSP–model into a CLSP–model we must drop (4). The constraints

$$p_j q_{jt} \leq C_{m_j t} y_{jt} \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{12}$$

and

$$x_{jt} = y_{jt} \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{13}$$

are introduced instead. Note that setup costs are charged now in every period in which production takes place. Again, the resulting model can be drastically reduced. (5) and (6) are redundant. Moreover, we can eliminate the variables x_{jt} by substitution. Usually, $v_j = 0$ is assumed for all or some $j = 1, \ldots, J$ since we have large (time) buckets in mind. This makes (3) obsolete for these items. It should be clear that once we give up (4) the resulting model does not support

sequence decisions any more. A general statement about the relation between optimal objective function values for PLSP–instances and corresponding CLSP–instances cannot be made.

3 Complexity Considerations

Only very little theoretical research has been done on complexity issues[1] for lot sizing (and scheduling) (see [29] for a review). Optimizing the single–level CLSP is known to be NP–hard [4]. So must be optimizing the multi–level CLSP. The complexity of the single–level, single–machine DLSP is examined in [30, 31]. While the feasibility problem is polynomially solvable, the optimization problem is claimed to be NP–hard. In [5] it is pointed out that the presented proof of this claim is false. Nevertheless, a correct proof also given in [5] shows that the optimization problem is (strongly) NP–hard. So must be the multi–level DLSP.

Theoretical results for any variant of the CSLP or the PLSP are not published yet. But, as we have learned from experience, for most multi–level PLSP instances it is quite a formidable task to find even a feasible solution. Using heuristics to attack the PLSP–MM (and its extensions) thus seems to be an extremely good piece of advice.

4 Derived Parameters

Throughout this text it will be helpful to have some notation for certain information that can be derived from the parameters defined above. Also, evaluating the parameters for some values not directly given deepens the understanding which is a motivation on its own right.

To start with, let us analyze the gozinto–structure a bit more. We already have defined the set of immediate successors $\mathcal{S}_j$ of an item j. The set of immediate predecessors $\mathcal{P}_j$ of an item j can be defined as follows:

$$\mathcal{P}_j \stackrel{def}{=} \{i = 1, \ldots, J \mid j \in \mathcal{S}_i\} \qquad j = 1, \ldots, J \tag{14}$$

On the basis of this, the set of all successors $\bar{\mathcal{S}}_j$ and the set of all predecessors $\bar{\mathcal{P}}_j$, respectively, of an item j are given as

$$\bar{\mathcal{S}}_j \stackrel{def}{=} \mathcal{S}_j \cup \left\{k \in \bar{\mathcal{S}}_i \mid i \in \mathcal{S}_j\right\} \qquad j = 1, \ldots, J \tag{15}$$

and

$$\bar{\mathcal{P}}_j \stackrel{def}{=} \mathcal{P}_j \cup \left\{k \in \bar{\mathcal{P}}_i \mid i \in \mathcal{P}_j\right\} \qquad j = 1, \ldots, J. \tag{16}$$

[1] An introduction into the theory of complexity would need more space than this section and is out of the scope of this text. We refer to [12].

Producing one unit of item j triggers the production of preceding items. The internal demand id_{ji} for an item i caused by producing one item j is computed as

$$id_{ji} \stackrel{def}{=} \begin{cases} 1 & , \text{if } j = i \\ 0 & , \text{if } i \notin \bar{\mathcal{P}}_j \\ \sum\limits_{k \in \mathcal{P}_j} a_{kj} id_{ki} \; (= \sum_{k \in \mathcal{S}_i} a_{ik} id_{jk}) & , \text{otherwise} \end{cases} \tag{17}$$

for $j, i = 1, \ldots, J$ where it should be emphasized that the case $j = i$ does not indicate cyclic structures, but eases the notation.

The expression

$$dep_j \stackrel{def}{=} \begin{cases} 0 & , \text{if } \mathcal{P}_j = \emptyset \\ \max\limits_{i \in \mathcal{P}_j} \{v_i + dep_i\} & , \text{otherwise} \end{cases} \quad j = 1, \ldots, J \tag{18}$$

defines the depth of an item. The low level code llc_j of an item j, informally given in Chapter 2, can be restated more formally:

$$llc_j \stackrel{def}{=} \begin{cases} 0 & , \text{if } \mathcal{S}_j = \emptyset \\ 1 + \max\limits_{i \in \mathcal{S}_j} \{llc_i\} & , \text{otherwise} \end{cases} \quad j = 1, \ldots, J \tag{19}$$

The net requirement nr_j of an item j can be computed in an item–by–item stepwise manner. To describe the calculation we assume a technological ordering without loss of generality, i.e.

$$j < i \Rightarrow j \notin \bar{\mathcal{P}}_i \qquad j, i = 1, \ldots, J$$

must hold. By the way, note

$$j < i \Rightarrow llc_j \leq llc_i \qquad j, i = 1, \ldots, J$$

would be fine as well, but gives less freedom for labeling items. Remember that only acyclic gozinto–structures are taken into account. The adjacency matrix representation of a gozinto–structure can thus always be written in triangular form

$$(a_{ji})_{j,i=1,\ldots,J} = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ a_{21} & \ddots & & & \vdots \\ a_{31} & a_{32} & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ a_{J1} & a_{J2} & \cdots & a_{J(J-1)} & 0 \end{pmatrix}.$$

Furthermore, we make use of an auxiliary function

$$\chi_1(x) \stackrel{def}{=} \frac{x + \mid x \mid}{2} = \begin{cases} x & , \text{if } x \geq 0 \\ 0 & , \text{otherwise} \end{cases}. \tag{20}$$

The procedure to determine nr_j can now be given as in Table 3. Without loss of generality, we subsequently assume that every item has a positive net requirement.

$nr_j := \sum_{t=1}^{T} d_{jt}$ for all $j = 1, \ldots, J$.
for $j = 1$ to $j = J$

$$nr_j := \chi_1 \left(nr_j + \sum_{i \in S_j} a_{ji} nr_i - I_{j0} \right).$$

Table 3: A Method to Compute the Net Requirement

5 Another Point of View: Gozinto–Trees

For illustrating graphically which item causes what internal demand one can convert matrices of gozinto–factors into acyclic but general graphs. While these pictures are compact and easy to read, they misleadingly reflect the precedence relations for scheduling items.[2] A representation that directly reveals the precedence relations in a feasible schedule is gained by converting a general gozinto–graph into an assembly structure by copying nodes with more than one successor. Figure 1 gives an example (see also [1] where this point of view helps to solve problems with complex product structures). Note, both representations contain exactly the same information and thus are equivalent. We will use the term gozinto–tree to refer to the gozinto–structure when converted into an assembly structure. It should be clear that the gozinto–tree actually is a forest, if the gozinto–structure contains more than one end item.

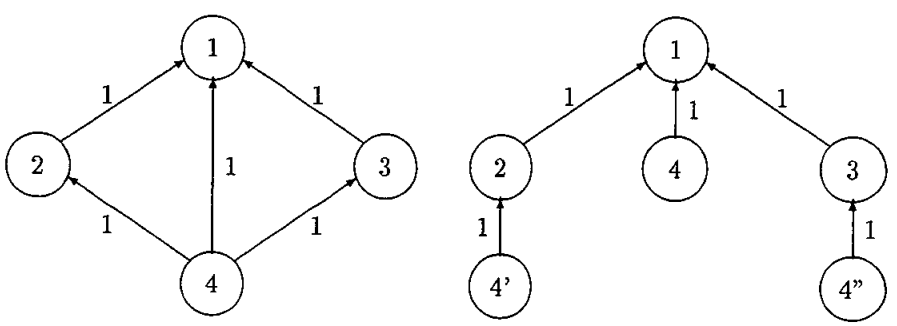

Figure 1: Converting a Gozinto–Structure into a Gozinto–Tree

Since gozinto–trees will play a key role in the methods to be presented, we

[2] Compare the graphical representation of a gozinto–structure with an activity–on–node network as used in project scheduling. In the latter one, precedence relations in the graph correspond to precedence relations that must be respected in every feasible schedule.

give here a formal definition of a gozinto–tree. Assume a matrix of gozinto–factors to be given. Then, a set denoted as Γ_j represents the gozinto–tree with item j being its root node. This set is recursively defined by

$$\Gamma_j \stackrel{def}{=} \{(j, \bigcup_{i \in \mathcal{P}_j} \Gamma_i)\} \qquad j = 1, \ldots, J. \tag{21}$$

The pair in the set Γ_j represents the root node in the corresponding gozinto–(sub–)tree. At the first position, we have the number of the item. At position number two, all immediately preceding nodes are shown. E.g., consider the gozinto–tree given in Figure 1:

$$\Gamma_1 = \{(1, \{(2, \{(4, \emptyset)\}), (3, \{(4, \emptyset)\}), (4, \emptyset)\})\}$$

6 Initial Inventory

In the single–level case one can assume without loss of generality that the initial inventory I_{j0} is zero for all items $j = 1, \ldots, J$. The reason for this is that positive stock levels can be mapped to external demands before the planning process begins. The rule to do so is very simple. Initial inventory of an item j is used to fulfill external demand in a period t only if all external demands in periods τ where $1 \leq \tau < t$ are also met by the initial inventory. More formally, $I_{j0} > \sum_{\tau=1}^{t-1} d_{j\tau}$ must hold if external demand in period t should be met by initial inventory. In the case that $I_{j0} \geq \sum_{t=1}^{T} d_{jt}$ holds, item j need not be produced at all. It is trivial to see that the preprocessed instance is feasible if and only if the original instance is feasible. And, the optimum objective function value of the preprocessed instance equals the optimum objective function value of the original instance.

In the multi–level case there are some strings attached which do not keep things that easy. A small example certainly brings this out best. Suppose $J = 2$ items are manufactured on a single machine. The gozinto–structure is given in Figure 2. Let the planning horizon be $T = 4$. All relevant data are given in Table 4. Points worth to be highlighted are that the machine is initially set up for item 1 ($y_{10} = 1$) and that there is a positive initial inventory for item 2 ($I_{20} = 10$).

Now assume that we proceed as we would do in the single–level case. That is, before we start with the planning process we eliminate the initial inventory during a preprocessing phase. Since the initial inventory meets the external demand for item 2 in period 2, the result is an instance without initial inventory where external demand for item 1 is left only. The optimum solution for the remaining instance is defined by a lot–for–lot policy ($q_{23} = q_{14} = d_{14} = 10$). The optimum objective function value is $s_1 + s_2 + h_2 I_{20} + h_2 a_{21} q_{14} = 1,900$.

The optimum solution for the unmodified instance is given in Table 5. Its objective function value is $s_2 + h_2(I_{20} - a_{21}q_{11}) + h_1(T-1)q_{11} + h_1(T-2)q_{12} =$

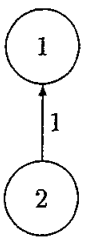

Figure 2: A Gozinto–Structure with Two Items

d_{jt}	$t=1$	2	3	4	h_j	s_j	p_j	v_j	y_{j0}	I_{j0}
$j=1$				10	20	900	1	1	1	
$j=2$		10			10	800	1	1		10
C_{1t}	15	15	15	15						

Table 4: Parameters of the Example

$1,350 < 1,900$.

q_{jt}	$t=1$	2	3	4
$j=1$	5	5		
$j=2$		10		
j with $y_{jt}=1$	1	2	2	2

Table 5: An Optimum Solution of the Example

Apparently, something goes wrong when eliminating initial inventory to get an instance with zero initial inventory. As pointed out for the single–level case, initial inventory of an item j can be mapped to demand in a period t without harm only if all other demand for item j in earlier periods can also be met by the initial inventory. In the multi–level case we have external as well as internal demand. But only the external demand is known by time. The time when internal demand occurs is not known unless a production plan is constructed. Hence, in general there cannot be a preprocessing procedure that eliminates initial inventory. As a consequence, multi–level lot sizing and scheduling procedures must take initial inventory into account.

Eventually, a rather unexpected phenomenon should be mentioned which

may occur if positive initial inventory comes along with multi–level gozinto–structures. Surprisingly, initial inventory may be the reason for producing items without any demand for it. Consider, for instance, the example above, this time, assuming $d_{jt} = 0$ for $j = 1, \ldots, J$ and $t = 1, \ldots, T$. In an optimum solution no production takes place. The optimum objective function value, however, is not zero, but $h_2 I_{20} T = 400$ for keeping 10 units of item 2 four periods in inventory. Suppose, $h_1 < h_2$. Then, the optimum result would be to produce $q_{11} = 10$ units of item 1 in period 1 which uses the initial inventory of item 2 up. Since there exists no demand for item 1, we have $I_{1T} = 10$, and the optimum objective function value is $h_1 q_{11} T < h_2 I_{20} T$.

7 Schedules

Discovering certain properties that optimum solutions must have allows to reduce the set of feasible solutions among which an optimum solution is to be and among which we should try to find a good feasible solution. We will now show that due to the combination of multi–level gozinto–structures and general holding costs a "nice" property of optimum schedules is no longer valid, though it is in the single–level case.

To avoid unnecessary repetitions let us assume once and for all the following parameters in this section: $M = 1$, $T = 10$, $C_{1t} = 15$, $p_j = 1$, $I_{j0} = 0$, and $v_j = 1$ for all $j = 1, \ldots, J$ and $t = 1, \ldots, T$. J will vary. The parameters y_{j0} and s_j are of no importance here and can be chosen arbitrarily.

For the first example which reveals a property of optimum schedules in the single–level case, let $J = 3$. External demand is given in Table 6. The holding cost $h_j > 0$ can be chosen arbitrarily for all $j = 1, \ldots, J$.

d_{jt}	$t=1$	2	3	4	5	6	7	8	9	10
$j=1$					20					20
$j=2$		20								
$j=3$									20	

Table 6: External Demand

Figure 3–(a) shows a feasible schedule. Note, the lot for item 1 fulfills the external demand in period 5 as well as the external demand in period 10. Suppose, that the sequence of lots should not be changed and wonder if there may exist another schedule (with the same sequence of lots and therefore the same total setup costs) with a lower objective function value (which means with lower total holding costs). The answer is yes, and the schedule with lowest objective function value (while respecting the sequence of lots) is depicted in Figure 3–(b). It is essential to understand that idle periods do not enforce additional setups.

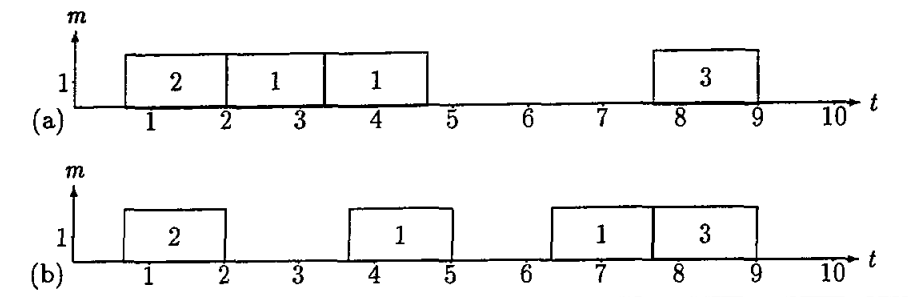

Figure 3: Two Schedules with the Same Sequence of Lots

The difference between both schedules is that in case (b) all items are shifted right–most without changing the sequence and without violating due dates. In general, given a sequence of lots to be produced on a common machine, the (unique) schedule with the lowest objective function value is the one which schedules all items right–most. Adopting the terminology of scheduling theory [32] here, such schedules are called semi–active schedules.[3] As we will now show, the property that an optimum schedule must be a semi–active one is valid for single–level structures only unless additional (restrictive) assumptions are made.

So, let us have a look at another example with $J = 2$ and a two–level gozinto–structure (see Figure 2). External demand is given in Table 7. Suppose here that $h_2 > h_1$ holds.

d_{jt}	$t=1$	2	3	4	5	6	7	8	9	10
$j=1$					20					20
$j=2$										

Table 7: External Demand

Figure 4–(a) provides a feasible schedule. For the given sequence of lots this is the unique semi–active schedule. Figure 4–(b), however, shows a schedule with lower objective function value respecting the sequence of lots. This one is not semi–active.

The reason why the second schedule has a lower objective function value is that holding item 2 is more expensive than holding item 1. All lots for item 1 should therefore be scheduled close to the lot of item 2 which meets the internal

[3] In contrast to our interpretation of semi–active schedules as right–most schedules, in scheduling theory left–most schedules are usually meant. The reason for this is a difference in the objective function. Minimizing the makespan is, for instance, a widespread objective.

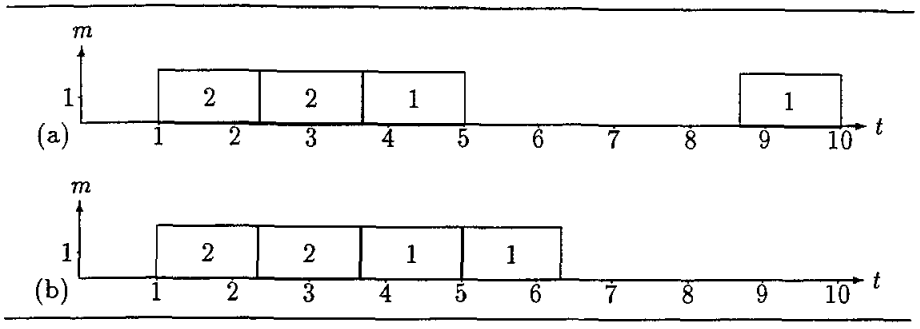

Figure 4: Two Schedules with the Same Sequence of Lots

demand. In general, a sufficient condition for an optimum schedule to be semi–active is

$$h_j \geq \sum_{i \in \mathcal{P}_j} a_{ij} h_i \qquad j = 1, \ldots, J \tag{22}$$

which is a restrictive assumption not made here (see Section 3.1). Nevertheless, as a result we have the fact that given a certain sequence of lots, those items j which fulfill (22) should be scheduled right–most.

8 Lot Splitting

Looking at all examples given up to here again, one might conclude that lot sizing (and scheduling) is about grouping demands together (and finding a (sub–)optimal schedule for these lots). Thus, lot sizes appear to be the sum of order sizes. While this is true for single–level, uncapacitated problems [36], lot sizing with multi–level gozinto–structures and/or capacity constraints needs to take splitting lots — splitting order sizes would be more precise[4] — into account and is therefore much more difficult as we will point out. The problem of lot sizing (and scheduling) where only grouping is allowed but no splitting, is called a batching (and sequencing) problem [18, 35, 37].[5]

[4]Note, lot splitting in our context does not necessarily mean to divide a lot into sublots and to process each sublot in parallel on identical facilities [2]. Sublots may also be processed consecutively (with other lots in–between).

[5]To convert the lot sizing problem into a batching problem we need to add additional constraints to the PLSP–MM–model formulation. If we assume $I_{j0} = 0$ for all $j = 1, \ldots, J$, the restrictions

$$I_{j(t-1)} - \sum_{i \in \mathcal{S}_j} \sum_{\tau=t}^{\min\{t+v_j-1,T\}} a_{ji} q_{i\tau} \leq B(1 - x_{jt}) \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix}$$

Let us investigate the uncapacitated, multi–level case first.[6] Assume $M = 2$ and a gozinto–structure as given in Figure 2. Table 8 gives the parameters that are of interest. Note that we have positive initial inventory. Furthermore, assume $h_2 > h_1$.

d_{jt}	$t=1$	2	3	m_j	I_{j0}	v_j
$j=1$			20	1		1
$j=2$				2	10	1

Table 8: Parameters of an Example without Capacity Constraints

The optimum production plan is depicted in Figure 5. It is remarkable to see, that the external demand of item 1 is split into two lots of size 10. Together with the results in Subsection 6 we now have that if generality should not be lost, multi–level gozinto–structures imply that lot splitting must be taken into account in order to find optimum solutions.

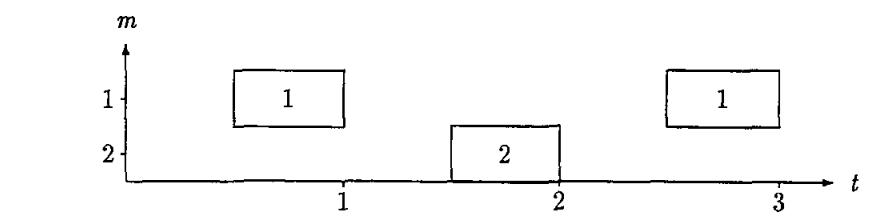

Figure 5: Lot Splitting in Uncapacitated Cases

Noteworthy to say that in capacitated cases lot splitting is not a problem introduced with multi–level structures. Suppose a single–level problem instance where $M = 1$, $T = 3$, and all other relevant parameters are given in Table 9. Setup costs are of no relevance. Note that the machine is initially set up for item 2. Let holding costs be arbitrarily chosen with respect to $h_1 > h_2$. There is only one feasible solution without lot splitting (see Figure 6–(b)). As we can see in Figure 6–(a), a feasible solution with lot splitting would decrease the objective function value by $10(h_1 - h_2) > 0$.

where B is a large number would work fine. Since batching is not in our interest, we omit further comments and leave the discussion of the case of positive initial inventory out.

[6] More formally, assume the problem under concern be defined by the PLSP–MM–model without the capacity constraints (7). The capacity parameters in (6) must be replaced with a large number, and all p_j–values can be replaced by one.

d_{jt}	$t=1$	2	3	p_j	y_{j0}	I_{j0}
$j=1$		20		1		
$j=2$			30	1	1	
C_{1t}	20	20	20			

Table 9: Parameters of the Example

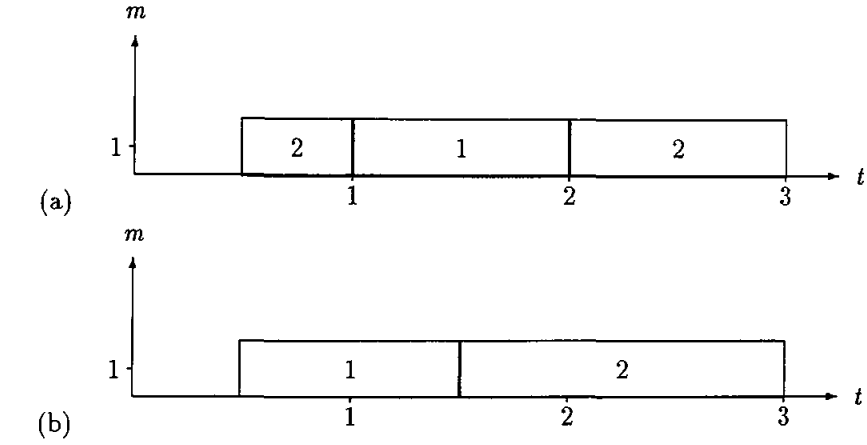

Figure 6: Schedules with and without Lot Splitting

A stronger observation is that some instances do have feasible solutions with lot splitting only. Again, suppose the parameters in Table 9 with slightly modified data, i.e. $d_{21} = 10$ and $y_{20} = 0$. To understand the logic of the example, it is important to be aware of the fact that the machine is not set up for any item initially. Figure 7 shows the unique feasible solution.

It is interesting to see that there are instances for which all feasible solutions are without lot splitting and no feasible solution with lot splitting exists. Once more, suppose the parameters given in Table 9 with slightly modified data, i.e. $d_{12} = 25$ and $y_{210} = 0$. Then, there is only one semi-active feasible solution which is depicted in Figure 8 where no lot splitting occurs.

In general, we have the following results.

- If there exists a feasible solution for an instance without lot splitting, the existence of feasible solutions with lot splitting is not guaranteed.
- If there exists a feasible solution for an instance with lot splitting, the existence of feasible solutions without lot splitting is not guaranteed, too.

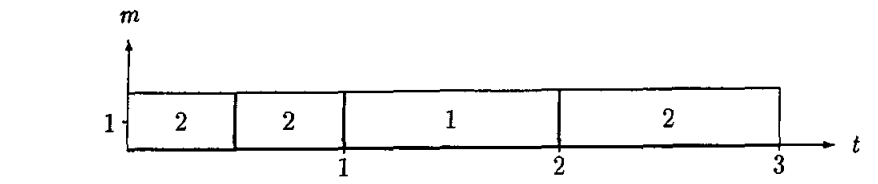

Figure 7: The Unique Feasible Solution of the Example

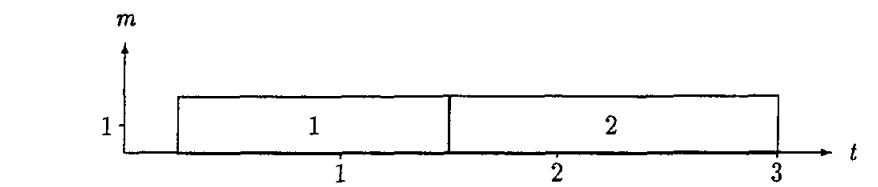

Figure 8: The Optimum Solution of the Example

- If both feasible solutions with and without lot splitting exist, the optimum solution is not necessarily without lot splitting.

Instances which have both types of feasible solutions are guaranteed to have an optimum solution without lot splitting if

$$h_j p_j \leq h_i p_i \qquad j, i = 1, \ldots, J \tag{23}$$

holds. In other words, we would have to assume that

$$h_j p_j = h_i p_i \qquad j, i = 1, \ldots, J \tag{24}$$

holds, to guarantee that for any instance having a feasible solution in which no lot splitting occurs an optimum solution can be found without lot splitting. As already stated above, we do not assume this here.

9 Some Valid Constraints

In this section we present some additional constraints which are proven to be valid in the sense that optimum solutions fulfilling these do exist. If there are any feasible solutions of the original PLSP–MM–model violating the new constraints, the valid constraints reduce the solution space and thus help to find

(sub–)optimum solutions (e.g. when standard MIP–solvers are employed or heuristics are to be developped) or at least help to gain a better understanding.

Valid constraint 1: The following equations determine the inventory at the end of each period. They can be used to eliminate the inventory variables.

$$I_{jt} = I_{j0} + \sum_{\tau=1}^{t} \left(q_{j\tau} - d_{j\tau} - \sum_{i \in S_j} a_{ji} q_{i\tau} \right) \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{25}$$

Proof: Trivial. It is a recursive evaluation of (2). □

Valid constraint 2: If there is no initial inventory, items must be produced only if there is a positive requirement.[7] This may fix the value of some decision variables to zero.

$$q_{jt} \leq nr_j \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{26}$$

$$x_{jt} \leq nr_j \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{27}$$

$$y_{jt} \leq nr_j \qquad \begin{matrix} j = 1, \ldots, J \\ t = 1, \ldots, T \end{matrix} \tag{28}$$

Proof: Trivial.[8] □

Valid constraint 3: For each item $j = 1, \ldots, J$ with $nr_j > 0$ the machine m_j must be set up for this item at least once. This holds for all feasible solutions.

$$nr_j \sum_{t=0}^{T} y_{jt} \geq nr_j \qquad j = 1, \ldots, J \tag{29}$$

Proof: Trivial. □

Valid constraint 4: Assume that $q_{jt} > 0$ should imply $q_{jt} \geq 1$ for $j = 1, \ldots, J$ and $t = 1, \ldots, T$. A setup for an item j where $j = 1, \ldots, J$ should only take place if the internal demand is met so that at least one item can be produced. This enforces some setup variables to be zero.

$$a_{ij} x_{jt} \leq I_{i(t-1)} + I_{it} \qquad \begin{matrix} j = 1, \ldots, J \\ i \in \mathcal{P}_j \\ t = 1, \ldots, T \end{matrix} \tag{30}$$

Proof: Let us consider items j and i and period t. If there is an optimum solution with $x_{jt} = 0$ the resulting inequality is true because of (9). So, assume there is an optimum solution with $x_{jt} = 1$. Due to (5) we have $y_{j(t-1)} = 0$ and $y_{jt} = 1$.

[7] Remember Subsection 6 where it is shown that the following unequalities are not necessarily true when we face positive initial inventory.

[8] Note, to let the constraints for setup state variable y_{jt} be valid (4) must not contain an equals sign.

Using (6) either $q_{jt} > 0$ or $q_{j(t+1)} > 0$ (if $t < T$) or both must hold, otherwise $y_{jt} = 0$ and hence $x_{jt} = 0$ would give a lower objective function value or would enforce $x_{j(t+1)} = 1$ (if $t < T$) resulting in another optimum solution. Suppose, $q_{jt} > 0$. Then we are done, since $a_{ij}x_{jt} \overset{q_{jt}>0}{\leq} a_{ij}q_{jt} \overset{(3)}{\leq} I_{i(t-1)} \overset{(9)}{\leq} I_{i(t-1)} + I_{it}$. Suppose, $t < T$ and $q_{j(t+1)} > 0$. Analogously, we are done again, because $a_{ij}x_{jt} \overset{q_{j(t+1)}>0}{\leq} a_{ij}q_{j(t+1)} \overset{(3)}{\leq} I_{it} \overset{(9)}{\leq} I_{i(t-1)} + I_{it}$. □

Valid constraint 5: Similar to (30) we can formulate new constraints for production quantities and setup state variables which set these decision variables to zero in some cases.

$$a_{ij}q_{jt} \leq I_{i(t-1)} + I_{it} \qquad \begin{array}{l} j = 1, \ldots, J \\ i \in \mathcal{P}_j \\ t = 1, \ldots, T \end{array} \tag{31}$$

$$a_{ij}(y_{jt} - y_{j(t-1)}) \leq I_{i(t-1)} + I_{it} \qquad \begin{array}{l} j = 1, \ldots, J \\ i \in \mathcal{P}_j \\ t = 1, \ldots, T \end{array} \tag{32}$$

Proof: Just follow the lines of (30). □

Valid constraint 6: A machine should switch the setup state in period $t = 1, \ldots, T$ for producing an item $j = 1, \ldots, J$ only if items $i \in \mathcal{P}_j$ with $I_{i0} = 0$ have a chance to be manufactured in advance. This condition lets some setup state variables be zero.

$$y_{jt} \leq y_{j0} + \sum_{\tau=0}^{t} y_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ i \in \{k \in \mathcal{P}_j \mid I_{k0} = 0 \text{ and } m_k \neq m_j\} \\ t = 1, \ldots, T \end{array} \tag{33}$$

$$y_{jt} \leq \sum_{\tau=0}^{t} y_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ i \in \{k \in \mathcal{P}_j \mid I_{k0} = 0 \text{ and } m_k = m_j\} \\ t = 1, \ldots, T \end{array} \tag{34}$$

Proof: We only prove the second inequality, because the first basically follows the same ideas. The difference is that in the first case an initial setup state can be kept up. Let the items under consideration be j and i where $i \in \{k \in \mathcal{P}_j \mid I_{k0} = 0 \text{ and } m_k = m_j\}$. Suppose that $\sum_{\tau=0}^{t} y_{i\tau} = 1$. Due to (8) the resulting inequality is true. So, assume there is an optimum solution with $\sum_{\tau=0}^{t} y_{i\tau} = 0$. With (6) we get $q_{it} = 0$ for all $t = 1, \ldots, t$. Because $I_{i0} = 0$, we also have $I_{i(t-1)} = I_{it} = 0$ using (2). (3) implies $q_{jt} = q_{j(t+1)} = 0$. Thus, there is an optimum solution with $y_{jt} = 0$. □

Valid constraint 7: Similar to the setup state variables we can formulate constraints for the setup variables to enforce some of these to be zero.

$$x_{jt} \leq y_{i0} + \sum_{\tau=1}^{t} x_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ i \in \{k \in \mathcal{P}_j \mid I_{k0} = 0\} \\ t = 1, \ldots, T \end{array} \tag{35}$$

Proof: Note, $y_{i0} + \sum_{\tau=1}^{t} x_{i\tau} = 0$ implies $\sum_{\tau=0}^{t} y_{i\tau} = 0$ due to (5) for all $i = 1, \ldots, J$. From then on the proof is similar to (34). □

Valid constraint 8: If there exists no initial inventory and if there is no future demand for an item $j = 1, \ldots, J$ there is no need to keep up the setup state or to produce anything. Again, this restricts the value of some decision variables

$$q_{jt} \leq \sum_{\tau=t}^{T} d_{j\tau} + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j}^{T} a_{ji} q_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ t = 1, \ldots, T \end{array} \tag{36}$$

$$x_{jt} \leq \sum_{\tau=t}^{T} d_{j\tau} + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j}^{T} a_{ji} q_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ t = 1, \ldots, T \end{array} \tag{37}$$

$$y_{jt} \leq \sum_{\tau=t}^{T} d_{j\tau} + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j}^{T} a_{ji} q_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ t = 1, \ldots, T \end{array} \tag{38}$$

where the latter two inequalities assume that $q_{jt} > 0$ implies $q_{jt} \geq 1$ for $j = 1, \ldots, J$ and $t = 1, \ldots, T$.

Proof: Let j be the item under concern and period t be the focus of attention. If the right hand side is positive, the result is trivial due to (8). So, assume there is an optimum solution with $\sum_{\tau=t}^{T} d_{j\tau} + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j}^{T} a_{ji} q_{i\tau} = 0$. Due to (2) in combination with (1) we get $q_{j\tau} = 0$ for all $\tau = t, \ldots, T$. With (6) there is an optimum solution with $y_{j\tau} = 0$ for all $\tau = t, \ldots, T$ which in turn implies $x_{j\tau} = 0$ for all $\tau = t, \ldots, T$ using (5). □

Valid constraint 9: Quite similar is a constraint which fixes setup state variables to zero if there is no future demand that is not met and if there is no initial inventory. Assume that if demand occurs, there is demand for at least one unit of an item.

$$\begin{aligned} y_{jt} \leq \; & y_{j(t+1)} \\ & + \sum_{\tau=t}^{T} d_{j\tau} \\ & + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j}^{T} a_{ji} q_{i\tau} \\ & - \sum_{\tau=t+2}^{T} q_{j\tau} \end{aligned} \qquad \begin{array}{l} j = 1, \ldots, J \\ t = 1, \ldots, T-1 \end{array} \tag{39}$$

Proof: Let us concern about item j and period t. First, analyze the structure of the right hand side:

$$\underbrace{y_{j(t+1)}}_{(I)} + \underbrace{\sum_{\tau=t}^{t+1} d_{j\tau} + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j}^{t+v_j+1} a_{ji} q_{i\tau}}_{(II)}$$
$$+ \underbrace{\sum_{\tau=t+2}^{T} d_{j\tau} + \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+v_j+2}^{T} a_{ji} q_{i\tau} - \sum_{\tau=t+2}^{T} q_{j\tau}}_{(III)}$$

Part (I) represents the setup state in period $t+1$. Part (II) evaluates to a positive value if there is any demand in period t or period $t+1$. Eventually, part (III) represents future demand that is not met. If the right hand side is positive, the result is trivial due to (8). So, assume an optimum solution where the right hand side evaluates to zero. Since $y_{j(t+1)} = 0$ no additional setup occurs in period t if $y_{jt} = 0$, too. For having no demand in periods t and $t+1$ to be met with production in t and $t+1$, respectively, and no future demand in periods $t+2$ to T which is not met an optimum solution would have $q_{jt} = q_{j(t+1)} = 0$ due to (2) in combination with (1). Hence, there is an optimum solution with $y_{jt} = 0$. □

10 Postprocessing

Given a particular instance, our goal is to heuristically find a (feasible) production plan with (sub–)optimum objective function value where finding a production plan means to determine the lot sizes and a schedule for the lots. Calculating the corresponding objective function value of a production plan is an easy task then.

In this section we will learn that once we have found a feasible solution represented by the matrix of production quantities q_{jt}, there are some degrees of freedom for the schedule. More precisely, a q_{jt}–matrix does not define a unique schedule — in our PLSP–MM–model formulation we thus introduced the setup state variables y_{jt} which determine a schedule.

A small (single–level) example should make things more clear. Suppose $J = 2$, $M = 1$, $T = 5$ and feasible production quantities as given in Table 10. Let the machine initially be set up for none of the items. Furthermore, assume $s_1 > s_2$. Since we do not need to know all the other parameters, they are not given here. Figure 9 shows three alternative feasible schedules.

Note, all schedules differ only in the sequence of lots within the periods. Assuming the production quantities to be feasible, all three Gantt–charts thus define feasible production plans. Also note that holding costs are determined with the matrix of production quantities. Hence, all three schedules cause the

q_{jt}	$t=1$	2	3	4	5
$j=1$		10		10	
$j=2$		10		10	

Table 10: A Matrix of Feasible Production Quantities

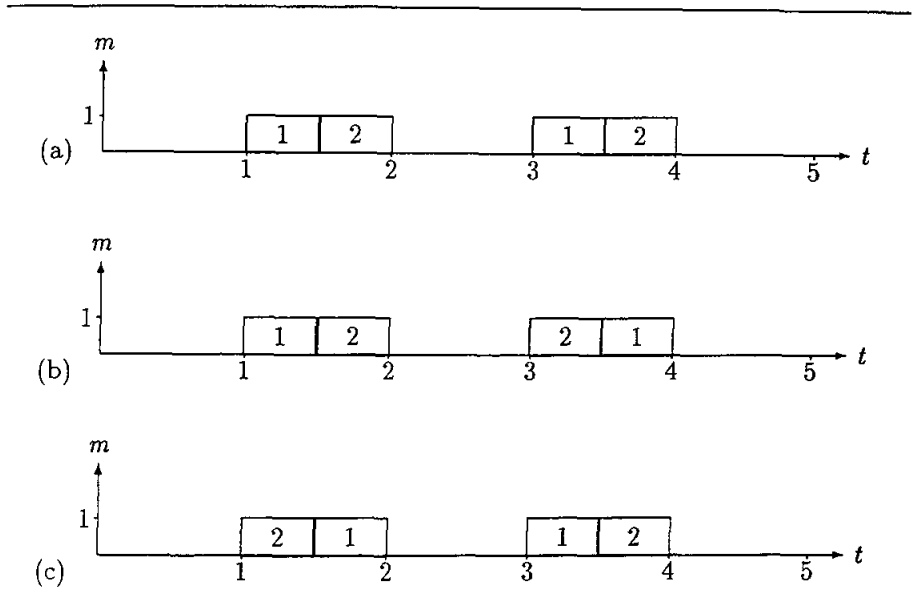

Figure 9: Three Feasible Schedules

same holding costs. But, if we consider setup costs, we can see that all three schedules cause different setup costs, namely, $2s_1 + 2s_2$ for schedule (a), $2s_1 + s_2$ for schedule (b), and $s_1 + 2s_2$ for schedule (c). By the way, no other schedule that corresponds to the quantities in Table 10 has a lower objective function value than schedule (c).

In general, we like to have a postprocessor which, once that we have found a feasible solution, determines a schedule with minimum total setup costs. o guarantee that the resulting schedule is feasible, the postprocessor should not modify the matrix of production quantities.[9] Under this assumption postprocessing the schedules for different machines does not interact. Basically, such a

[9]A formal statement of this problem emerges from the PLSP-MM-model formulation if production quantities q_{jt} are used as parameters and not as decision variables.

module could enumerate all sequences of lots within a period for all machines, thus enumerating $\sum_{m=1}^{M} 2^{\check{T}_m}$ schedules where

$$\check{T}_m \stackrel{def}{=} \sum_{t=1}^{T} \chi_2 \left(-1 + \sum_{j \in \mathcal{J}_m} \chi_2(q_{jt}) \right)$$

for all $m = 1, \ldots, M$ and χ_2 is an auxiliary function defined as

$$\chi_2(x) \stackrel{def}{=} \begin{cases} 1 & \text{, if } x > 0 \\ 0 & \text{, otherwise} \end{cases} . \tag{40}$$

Subsequently, we present a more efficient procedure.

We give here a functional specification of a postprocessor Ξ which, when applied to each machine, moves from period 1 to period t in a single pass and constructs the schedule with minimum total setup costs making use of a feasible q_{jt}–matrix. The output will be represented as a list of item indices. Each entry in the list corresponds to a positive entry in the matrix of production quantities. Hence, we have

$$\sum_{j \in \mathcal{J}_m} \sum_{t=1}^{T} \chi_2(q_{jt})$$

entries in the list representing the schedule for machine m. Reading the list from left to right uniquely defines a sequence of production quantities (and in combination with the q_{jt}–matrix a schedule, of course). For instance, applying the postprocessing phase to the example above should give $< 2 : 1 : 1 : 2 >_1$ representing the schedule for machine 1 in Figure 9–(c).[10]

Before we can give a formal definition of the postprocessor, let us introduce some notation for the sake of convenience. If machine m and period t are under concern, we have to evaluate $\sum_{j \in \mathcal{J}_m} \chi_2(q_{jt})$ using Σ_{mt} as a short–hand notation for it. Note, since we consider the PLSP where at most one setup may take place per period, $\Sigma_{mt} \in \{0, 1, 2\}$. If $\Sigma_{mt} = 1$, let j^* be the unique item with $q_{j^*t} > 0$. If $\Sigma_{mt} = 2$, let $j_1^* < j_2^*$ be the items with non–zero production quantities. Furthermore, let j_{m0} be the item machine m is initially setup for, or $j_{m0} = 0$ if the machine m is set up for no item initially.

To apply the postprocessor Ξ to machine m, just evaluate

$$\Xi(m) \stackrel{def}{=} \Xi_1^m(j_{m0}, 1). \tag{41}$$

The function Ξ_1^m handles the case in which a period t is under concern and the setup state at the beginning of this period is fixed. This certainly happens

[10]The formal definition of the postprocessor will need some operations on lists. We use $<>_m$ to denote an empty list, and a $\circ$–symbol to denote concatenation. For example, $< a : \ldots : z >_m \circ <>_m = < a : \ldots : z >_m$ and $< a : \ldots : w >_m \circ < x : y : z >_m = < a : \ldots : z >_m$.

to be in period 1 where the setup state at the beginning of that period is j_{m0}. Five cases may occur:
Case 1: $t > T$
Case 2: $\Sigma_{mt} = 0$ and $t \leq T$
Case 3: $\Sigma_{mt} = 1$ and $j^* = j$ and $t \leq T$
Case 4: $\Sigma_{mt} = 1$ and $j^* \neq j$ and $t \leq T$
Case 5: $\Sigma_{mt} = 2$ and $j_i^* = j$ $(i \in \{1,2\})$ and $t \leq T$

Depending on the case that holds, Ξ_1^m is defined as follows:

$$\Xi_1^m(j,t) \stackrel{def}{=} \begin{cases} <> & \textit{Case 1} \\ \Xi_2^m(j,t+1) & \textit{Case 2} \\ < j > \circ\, \Xi_2^m(j^*,t+1) & \textit{Case 3} \\ < j^* > \circ\, \Xi_1^m(j^*,t+1) & \textit{Case 4} \\ < j : j^*_{(3-i)} > \circ\, \Xi_1^m(j^*_{(3-i)},t+1) & \textit{Case 5} \end{cases} \tag{42}$$

First, there is the terminal case when all periods are considered. Second, if no production takes place in period t on machine m, no entry is added to the list which defines the schedule. A recursive call to Ξ_2^m passes the information that item j is the last item the machine is setup for. Third, there is the case where only one item j is produced on machine m in period t and machine m is already setup for this item. We then add the item index j to the list and proceed on with a call to Ξ_2^m. The fourth case is similar to the third, but this time the machine is not in the proper setup state. We thus add j^* to the list and call Ξ_1^m, because at the beginning of period $t+1$ the machine is now set up for item j^*. Finally, if there are two items to be produced, one of them must be item j. Otherwise, we would have two setups in period t which is not allowed. The sequence in which these items are produced must therefore have item j at its first position. At the end of period t and thus at the beginning of period $t+1$, the machine is set up for the second item then.

The function Ξ_2^m handles the case in which a period t is under concern and the setup state at the beginning of that period is not fixed yet. Again, we have five cases to distinguish.
Case 1: $t > T$
Case 2: $\Sigma_{mt} = 0$ and $t \leq T$
Case 3: $\Sigma_{mt} = 1$ and $t \leq T$
Case 4: $\Sigma_{mt} = 2$ and $j_i^* = j$ $(i \in \{1,2\})$ and $t \leq T$
Case 5: The above cases do not hold.

Ξ_2^m is defined as follows:

$$\Xi_2^m(j,t) \stackrel{def}{=} \begin{cases} <> & \textit{Case 1} \\ \Xi_2^m(j,t+1) & \textit{Case 2} \\ < j^* > \circ\, \Xi_2^m(j^*,t+1) & \textit{Case 3} \\ < j : j^*_{(3-i)} > \circ\, \Xi_1^m(j^*_{(3-i)},t+1) & \textit{Case 4} \\ \hat{\Xi}_1^m(j_1^*,j_2^*,t+1) & \textit{Case 5} \end{cases} \tag{43}$$

The first one is the terminal case. Case number two deals with situations in which nothing is to produce on machine m in period t. We simply move on with a recursive call to Ξ_2^m then. In case three, which is the case when only one item is to be manufactured, the item j^* is added to the list that defines the schedule. More interesting is case four where two items are to be produced on machine m in period t and one of them is the last item the machine is set up for. If this holds, it is clear that item j is to be scheduled first, because otherwise the setup costs for item j would be charged twice. Last, it may happen that two items are to be produced, but none of them is the one the machine is already set up for. A call to $\hat{\Xi}_1^m$ finds out the right sequence.

The function $\hat{\Xi}_1^m$ handles the case in which period t is under concern and the setup state at the beginning of that period is fixed. In contrast to the function Ξ_1^m, we employ $\hat{\Xi}_1^m$ if we are not quite sure which item the machine should be set up for at the beginning of period t. Two alternatives do exist. In total we have seven cases to discriminate:

Case 1: $t > T$
Case 2: $\Sigma_{mt} = 0$ and $t \leq T$
Case 3: $\Sigma_{mt} = 1$ and $j^* = j_i$ $(i \in \{1,2\})$ and $t \leq T$
Case 4: $\Sigma_{mt} = 1$ and $j_1 \neq j^* \neq j_2$ and $t \leq T$
Case 5: $\Sigma_{mt} = 2$ and $j_i^* = j_i \wedge j_{(3-i)}^* \neq j_{(3-i)}$ $(i \in \{1,2\})$ and $t \leq T$
Case 6: $\Sigma_{mt} = 2$ and $j_1^* = j_1 \wedge j_2^* = j_2$ and $s_{j_1} \geq s_{j_2}$ and $t \leq T$
Case 7: $\Sigma_{mt} = 2$ and $j_1^* = j_1 \wedge j_2^* = j_2$ and $s_{j_1} < s_{j_2}$ and $t \leq T$

The function $\hat{\Xi}_1^m$ is defined as follows:

$$\hat{\Xi}_1^m(j_1, j_2, t) \stackrel{def}{=} \begin{cases} <> & \textit{Case 1} \\ \hat{\Xi}_2^m(j_1, j_2, t+1) & \textit{Case 2} \\ < j_{(3-i)} : j_i : j_i > \circ\, \Xi_2^m(j^*, t+1) & \textit{Case 3} \\ < j_1 : j_2 : j^* > \circ\, \Xi_1^m(j^*, t+1) & \textit{Case 4} \\ \quad < j_{(3-i)} : j_i : j_i : j_{(3-i)}^* > & \\ \quad \circ\, \Xi_1^m(j_{(3-i)}^*, t+1) & \textit{Case 5} \\ < j_2 : j_1 : j_1 : j_2 > \circ\, \Xi_1^m(j_2^*, t+1) & \textit{Case 6} \\ < j_1 : j_2 : j_2 : j_1 > \circ\, \Xi_1^m(j_1^*, t+1) & \textit{Case 7} \end{cases} \tag{44}$$

Many cases may occur. Of course, there is the terminal case. If nothing is to be produced in period t, we can still not be sure in which sequence the items that are kept in memory are to be scheduled. However, we know that the setup state of machine m at the end of period t is not fixed yet, and thus we call $\hat{\Xi}_2^m$. The third case deals with situations in which one item is to be produced in period t and in which this single item equals one of those that are remembered. Hence, we build a lot for that item which is the cheapest possible sequence. If the item that is to be produced is not equal to one of those that are pending to be added to the list, case four holds. The sequence in which the two pending items are scheduled can be chosen arbitrarily then. Case five is the case where two items are to be scheduled in period t, but only one of them equals one of the

pending items. The sequence in which the items are scheduled is the one which causes the least setup costs. Cases six and seven describe what to do if the two pending items are to be produced on machine m in period t again. A lot is then built for the item with the highest setup cost.

The function $\hat{\Xi}_2^m$ handles the case in which period t is under concern and the setup state at the beginning of that period is not fixed. Also, we face a pending sequence decision from earlier periods which is in difference to Ξ_2^m. The function $\hat{\Xi}_2^m$ is defined as follows:

$$\hat{\Xi}_2^m(j_1, j_2, t) \stackrel{def}{=} \begin{cases} \ldots & \text{, see } \textit{Cases 1 to 7} \text{ of } \hat{\Xi}_1^m \\ < j_1 : j_2 > & \\ \circ\ \hat{\Xi}_1^m(j_1^*, j_2^*, t+1) & \text{, otherwise} \end{cases} \tag{45}$$

This definition is almost equal to the definition of $\hat{\Xi}_1^m$, i.e. we have the same seven cases as before. The only difference is an eighth case which applies when two items are to be produced in period t and none of them equals any of the pending items.

To illustrate the formulae, let us run a postprocessing phase for the example above where production quantities are given in Table 10. Remember, $j_{10} = 0$ and $s_1 > s_2$.

$$\begin{aligned} \Xi(1) &= \Xi_1^1(0,1) \\ &= \Xi_2^1(0,2) \\ &= \hat{\Xi}_1^1(1,2,3) \\ &= \hat{\Xi}_2^1(1,2,4) \\ &= < 2:1:1:2 >_1 \circ \Xi_1^1(2,5) \\ &= < 2:1:1:2 >_1 \circ \Xi_2^1(2,6) \\ &= < 2:1:1:2 >_1 \circ <>_1 \\ &= < 2:1:1:2 >_1 \end{aligned}$$

This result indeed corresponds to the schedule given in Figure 9–(c).

Acknowledgement

This work was done with partial support from the DFG–project Dr 170/4–1.

References

[1] Afentakis, P., Gavish, B., (1986), Optimal Lot–Sizing Algorithms for Complex Product Structures, Operations Research, Vol. 34, pp. 237–249

[2] American Production and Inventory Control Society Inc., (1995), APICS Dictionary, Falls Church, 8th edition

[3] BITRAN, G.R., MATSUO, H., (1986), Approximation Formulations for the Single–Product Capacitated Lot Size Problem, Operations Research, Vol. 34, pp. 63–74

[4] BITRAN, G.R., YANASSE, H.H., (1982), Computational Complexity of the Capacitated Lot Size Problem, Management Science, Vol. 28, pp. 1174–1186

[5] BRÜGGEMANN, W., (1995), Ausgewählte Probleme der Produktionsplanung — Modellierung, Komplexität und neuere Lösungsmöglichkeiten, Physica–Schriften zur Betriebswirtschaft, Vol. 52, Heidelberg, Physica

[6] DIABY, M., BAHL, H.C., KARWAN, M.H., ZIONTS, S., (1992), A Lagrangean Relaxation Approach for Very–Large–Scale Capacitated Lot–Sizing, Management Science, Vol. 38, pp. 1329–1340

[7] DINKELBACH, W., (1964), Zum Problem der Produktionsplanung in Ein– und Mehrproduktunternehmen, Würzburg, Physica, 2nd edition

[8] DREXL, A., HAASE, K., (1995), Proportional Lotsizing and Scheduling, International Journal of Production Economics, Vol. 40, pp. 73–87

[9] EPPEN, G.D., MARTIN, R.K., (1987), Solving Multi–Item Capacitated Lot–Sizing Problems Using Variable Redefinition, Operations Research, Vol. 35, pp. 832–848

[10] FLEISCHMANN, B., (1990), The Discrete Lot–Sizing and Scheduling Problem, European Journal of Operational Research, Vol. 44, pp. 337–348

[11] 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

[12] GAREY, M.R., JOHNSON, D.S., (1979), Computers and Intractability: A Guide to the Theory of NP–completeness, San Francisco, Freeman

[13] HAASE, K., (1993), Capacitated Lot–Sizing with Linked Production Quantities of Adjacent Periods, Working Paper No. 334, University of Kiel

[14] HAASE, K., (1994), Lotsizing and Scheduling for Production Planning, Lecture Notes in Economics and Mathematical Systems, Vol. 408, Berlin, Springer

[15] HAASE, K., KIMMS, A., (1996), Lot Sizing and Scheduling with Sequence Dependent Setup Costs and Times and Efficient Rescheduling Opportunities, Working Paper No. 393, University of Kiel

[16] HINDI, K.S., (1996), Solving the CLSP by a Tabu Search Heuristic, Journal of the Operational Research Society, Vol. 47, pp. 151–161

[17] VAN HOESEL, S., KOLEN, A., (1994), A Linear Description of the Discrete Lot–Sizing and Scheduling Problem, European Journal of Operational Research, Vol. 75, pp. 342–353

[18] JORDAN, C., (1995), Batching and Scheduling — Models and Methods for Several Problem Classes, Ph.D. dissertation, University of Kiel

[19] KARMARKAR, U.S., KEKRE, S., KEKRE, S., (1987), The Deterministic Lotsizing Problem with Startup and Reservation Costs, Operations Research, Vol. 35, pp. 389–398

[20] KARMARKAR, U.S., SCHRAGE, L., (1985), The Deterministic Dynamic Product Cycling Problem, Operations Research, Vol. 33, pp. 326–345

[21] KIMMS, A., (1994), Optimal Multi–Level Lot Sizing and Scheduling with Dedicated Machines, Working Paper No. 351, University of Kiel

[22] KIMMS, A., (1996), Multi–Level, Single–Machine Lot Sizing and Scheduling (with Initial Inventory), European Journal of Operational Research, Vol. 89, pp. 86–99

[23] KIMMS, A., (1996), Competitive Methods for Multi–Level Lot Sizing and Scheduling: Tabu Search and Randomized Regrets, International Journal of Production Research, Vol. 34, pp. 2279–2298

[24] KIMMS, A., (1996), Multi–Level Lot Sizing and Scheduling — Methods for Capacitated, Dynamic, and Deterministic Models, Ph.D. dissertation, University of Kiel

[25] KIRCA, Ö., KÖKTEN, M., (1994), A New Heuristic Approach for the Multi–Item Dynamic Lot Sizing Problem, European Journal of Operational Research, Vol. 75, pp. 332–341

[26] LASDON, L.S., TERJUNG, R.C., (1971), An Efficient Algorithm for Multi–Item Scheduling, Operations Research, Vol. 19, pp. 946–969

[27] LOTFI, V., CHEN, W.H., (1991), An Optimal Algorithm for the Multi–Item Capacitated Production Planning Problem, European Journal of Operational Research, Vol. 52, pp. 179–193

[28] MAES, J, VAN WASSENHOVE, L.N., (1988), Multi–Item Single–Level Capacitated Dynamic Lot–Sizing Heuristics: A General Review, Journal of the Operational Research Society, Vol. 39, pp. 991–1004

[29] POTTS, C.N., VAN WASSENHOVE, L.N., (1992), Integrating Scheduling with Batching and Lot-Sizing: A Review of Algorithms and Complexity, Journal of the Operational Research Society, Vol. 43, pp. 395–406

[30] SALOMON, M., (1991), Deterministic Lotsizing Models for Production Planning, Lecture Notes in Economics and Mathematical Systems, Vol. 355, Berlin, Springer

[31] SALOMON, M., KROON, L.G., KUIK, R., VAN WASSENHOVE, L.N., (1991), Some Extensions of the Discrete Lotsizing and Scheduling Problem, Management Science, Vol. 37, pp. 801–812

[32] SPRECHER, A., KOLISCH, R., DREXL, A., (1995), Semi–Active, Active, and Non–Delay Schedules for the Resource–Constrained Project Scheduling Problem, European Journal of Operational Research, Vol. 80, pp. 94–102

[33] TEMPELMEIER, H., DERSTROFF, M., (1996), A Lagrangean–Based Heuristic for Dynamic Multi–Level Multi–Item Constrained Lotsizing with Setup Times, Management Science, Vol. 42, pp. 738–757

[34] TEMPELMEIER, H., HELBER, S., (1994), A Heuristic for Dynamic Multi–Item Multi–Level Capacitated Lotsizing for General Product Structures, European Journal of Operational Research, Vol. 75, pp. 296–311

[35] UNAL, A.T., KIRAN, A.S., (1992), Batch Sequencing, IIE Transactions, Vol. 24, No. 4, pp. 73–83

[36] WAGNER, H.M., WHITIN, T.M., (1958), Dynamic Version of the Economic Lot Size Model, Management Science, Vol. 5, pp. 89–96

[37] WOODRUFF, D.L., SPEARMAN, M.L., (1992), Sequencing and Batching for Two Classes of Jobs with Deadlines and Setup–Times, Production and Operations Management, Vol. 1, pp. 87–102