Kimms, Alf; Drexl, Andreas

Working Paper — Digitized Version

Proportional lot sizing and scheduling: Some extensions

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 407

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

Suggested Citation: Kimms, Alf; Drexl, Andreas (1996) : Proportional lot sizing and scheduling: Some extensions, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 407, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/149038

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 407

Proportional Lot Sizing and Scheduling: Some Extensions

A. Kimms and A. Drexl

No. 407

Proportional Lot Sizing and Scheduling: Some Extensions

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 contribution generalizes the work of Drexl and Haase about the so-called proportional lot sizing and scheduling problem which was published in 1995. While the early paper considers single-level cases only, the paper at hand describes multi-level problems. It provides mixed-integer programs for several important extensions which differ in the allocation of resources. A generic solution method is presented and, following the preceding paper, a randomized regret based sampling method is tested. A computational study proves that, even for the multi-level case which is far more complex than the single-level problem, promising results are gained.

Keywords: Multi-level lot sizing, scheduling, PLSP, random sampling

1 Introduction

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) [6, 12, 21]. 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) [3, 9, 15, 23, 26] as well as the continuous setup lot sizing problem (CSLP) [1, 17, 16]. 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) [2, 7, 10, 14, 22, 24, 25] 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 [11]. In [13] a large bucket single–level lot sizing and scheduling model is discussed.

A comprehensive review of the multi–level lot sizing literature is given in [21] where it is shown that most authors do not take capacity restrictions into account and that they make restrictive assumptions such as linear or assembly gozinto–structures. If scarce capacities are considered, the work is mostly confined to single–machine cases. The most general methods are described in [27, 28] where the multi–level CLSP is attacked.

2 Multi–Level PLSP with Multiple Machines

An important variant of the PLSP is the one with multiple machines (PLSP–MM). 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 [19, 20], 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 [18].

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.

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

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

subject to

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

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

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

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

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

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

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

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

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

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. $\mathcal{J}_m \overset{def}{=} \{j \in \{1,\ldots,J\} \mid m_j = m\}$.
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. $\mathcal{S}_j \overset{def}{=} \{i \in \{1,\ldots,J\} \mid a_{ji} > 0\}$.
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

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 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.

3 Construction Principles

There is a generic construction scheme that forms the basis of our heuristic. It is a backward oriented procedure which schedules items period by period starting with period T and ending with period one. We choose here a recurrent representation which enables us to develop the underlying ideas in a stepwise fashion. Now, let us assume that *construct*$(t, \Delta t, m)$ is the procedure to be defined and $t + \Delta t$ is the period and m is the machine under concern. Again, $\Delta t \in \{0, 1\}$ where $\Delta t = 1$ indicates that the setup state for machine m at the beginning of period $t + 1$ is to be fixed next and $\Delta t = 0$ indicates that we already have chosen a setup state at the end of period t. The symbol j_{mt} will denote the setup state for machine m at the end of period t. Assume $j_{mt} = 0$ for $m = 1, \ldots, M$ and $t = 1, \ldots, T$ initially.

Note, from the problem parameters we can easily derive $\mathcal{P}_j$, the set of the immediate predecessors of item j, and $\bar{\mathcal{P}}_j$, the set of all predecessors of item j. Also, nr_j, the net requirement of item j, and id_{ji}, the internal demand for item i that is directly or indirectly caused by producing one unit of item j, are easy to compute.

Before the construction mechanism starts, the decision variables y_{jt} and q_{jt} are assigned zero for $j = 1, \ldots, J$, $m = 1, \ldots, M$, and $t = 1, \ldots, T$. Remember, given the values for y_{jt} and q_{jt} the values for x_{jt} and I_{jt} are implicitly defined. Furthermore, assume auxiliary variables $\tilde{d}_{jt}$ and CD_{jt} for $j = 1, \ldots, J$ and $t = 1, \ldots, T$. The former ones represent the entries in the demand matrix and thus are initialized with $\tilde{d}_{jt} = d_{jt}$. The latter ones stand for the cumulative future demand for item j which is not been met yet. As we will see, the cumulative demand can be efficiently computed while moving on from period to period. For the sake of convenience we introduce $CD_{j(T+1)} = 0$ for $j = 1, \ldots, J$. The remaining capacity of machine m in period t is denoted as RC_{mt}. Initially, $RC_{mt} = C_{mt}$ for $m = 1, \ldots, M$ and $t = 1, \ldots, T$.

The initial call is *construct*$(T, 1, 1)$ and initiates the fixing of setup states at the end of period T. Table 3 gives all the details.

```
choose $j_{mT} \in \mathcal{I}_{mT}$.
if ($j_{mT} \neq 0$)
    $y_{j_{mT}T} := 1$.
if ($m = M$)
    construct$(T, 0, 1)$.
else
    construct$(T, 1, m + 1)$.
```

Table 3: Evaluating *construct*$(T, 1, \cdot)$

The choice of j_{mT} needs to be refined, but at this point we do not need any further insight and suppose that the selection is done somehow. All we need to know is that $\mathcal{I}_{mt} \subseteq \mathcal{J}_m \cup \{0\}$ for $m = 1, \ldots, M$ and $t = 1, \ldots, T$ is the set of items among which items are chosen. Item 0 is a dummy item which will be needed for some methods that will be discussed. We will return for a precise discussion in subsequent sections. As one can see, once a setup state is chosen for all machines at the end of period T, a call of *construct*$(T, 0, 1)$ is made. Table 4 provides a recipe of how to evaluate such calls.

```
for $j \in \mathcal{J}_m$
        $CD_{jt} := \min\left\{CD_{j(t+1)} + \tilde{d}_{jt}, \max\{0, nr_j - \sum_{\tau=t+1}^{T} q_{j\tau}\}\right\}$.
if $(j_{mt} \neq 0)$
        $q_{j_{mt}t} := \min\left\{CD_{j_{mt}t}, \frac{RC_{mt}}{p_{j_{mt}}}\right\}$.
        $CD_{j_{mt}t} := CD_{j_{mt}t} - q_{j_{mt}t}$.
        $RC_{mt} := RC_{mt} - p_{j_{mt}}q_{j_{mt}t}$.
        for $i \in \mathcal{P}_{j_{mt}}$
                if $(t - v_i > 0$ and $q_{j_{mt}t} > 0)$
                        $\tilde{d}_{i(t-v_i)} := \tilde{d}_{i(t-v_i)} + a_{ij_{mt}}q_{j_{mt}t}$.
if $(m = M)$
        $construct(t-1, 1, 1)$.
else
        $construct(t, 0, m+1)$.
```

Table 4: Evaluating *construct*$(t, 0, \cdot)$ where $1 \leq t \leq T$

The situation when calling *construct*$(t, 0, m)$ is that the setup state j_{mt} has already been chosen. Remarkable to note, how easy it is to take initial inventory into account. This is due to the backward oriented scheme. Evaluating

$$\min\left\{CD_{j(t+1)} + \tilde{d}_{jt}, \max\{0, nr_j - \sum_{\tau=t+1}^{T} q_{j\tau}\}\right\} \tag{10}$$

makes sure that for an item j no more than the net requirement nr_j is produced. Note, cumulating the production quantities is an easy task which can be done very efficiently. Given the cumulative demand $CD_{j_{mt}t}$, production quantities $q_{j_{mt}t}$ can be determined with respect to capacity constraints. Afterwards, we simply update the $\tilde{d}_{jt}$–matrix to take internal demand into account and proceed. Table 5 describes how to evaluate *construct*$(t, 1, \cdot)$–calls.

These lines closely relate to what is defined in Table 4. Differences lie in the fact that a setup state is chosen for the end of period t but items are scheduled in period $t+1$. For computing production quantities we must therefore take into account that item $j_{m(t+1)}$ may already be scheduled in period $t+1$.

```
choose $j_{mt} \in \mathcal{I}_{mt}$.
if ($j_{mt} \neq 0$)
    $y_{j_{mt}t} := 1$.
    if ($j_{mt} \neq j_{m(t+1)}$)
        $q_{j_{mt}(t+1)} := \min\left\{CD_{j_{mt}(t+1)}, \frac{RC_{m(t+1)}}{p_{j_{mt}}}\right\}$.
        $CD_{j_{mt}(t+1)} := CD_{j_{mt}(t+1)} - q_{j_{mt}(t+1)}$.
        $RC_{m(t+1)} := RC_{m(t+1)} - p_{j_{mt}} q_{j_{mt}(t+1)}$.
        for $i \in \mathcal{P}_{j_{mt}}$
            if ($t + 1 - v_i > 0$ and $q_{j_{mt}(t+1)} > 0$)
                $\tilde{d}_{i(t+1-v_i)} := \tilde{d}_{i(t+1-v_i)} + a_{ij_{mt}} q_{j_{mt}(t+1)}$.
if ($m = M$)
    $construct(t, 0, 1)$.
else
    $construct(t, 1, m + 1)$.
```

Table 5: Evaluating *construct*$(t, 1, \cdot)$ where $1 \leq t < T$

Note, the combination of what is given in Tables 4 and 5 enforces that every item j_{mt} that is produced at the beginning of a period $t + 1$ is also produced at the end of period t if there is any positive cumulative demand left. In preliminary tests not reported here we also found out that if capacity is exhausted, i.e. if $RC_{m(t+1)} = 0$ and $CD_{j_{m(t+1)}(t+1)} > 0$, it is best to choose $j_{mt} = j_{m(t+1)}$ in Table 5. In other words, lots are not split.[1] The reason why this turned out to be advantageous is that the setup state tends to flicker otherwise and thus the total sum of setup costs tends to be high. In the rest of this chapter we assume that lots are not split.

Turning back to the specification of the *construct*-procedure, it remains to explain what shall happen when the first period is reached. Table 6 describes how to schedule those items in period 1 for which the machines are initially set up for. In contrast to what is given in Table 5 the initial setup state is known and thus needs not to be chosen.

A call to *construct*$(0, 0, \cdot)$ terminates the construction phase. What is left is a final feasibility test where

$$nr_j = \sum_{t=1}^{T} q_{jt} \tag{11}$$

must hold for $j = 1, \ldots, J$ for being a feasible solution. Eventually, the objective function value of a feasible solution can be determined.

[1]It is worth to be stressed that lot splitting could be easily integrated by *not* checking for exhausted capacity. All methods based on the described construction scheme may thus be adapted for lot splitting with minor modifications only.

```
if ($j_{m0} \neq j_{m1}$)
    $q_{j_{m0}1} := \min\left\{CD_{j_{m0}1}, \frac{RC_{m1}}{p_{j_{m0}}}\right\}$.
    $CD_{j_{m0}1} := CD_{j_{m0}1} - q_{j_{m0}1}$.
if ($m = M$)
    $construct(0, 0, 1)$.
else
    $construct(0, 1, m + 1)$.
```

Table 6: Evaluating $construct(0, 1, \cdot)$

To terminate a run of the construction procedure before period 1 is reached, we can perform a capacity check testing

$$\sum_{j \in \mathcal{J}_m} \sum_{i \in \{\bar{\mathcal{P}}_j \cup \{j\}\} \cap \mathcal{J}_m} p_i id_{ji} CD_{j(t+\Delta t)} > \sum_{\tau=1}^{t+\Delta t} C_{m\tau} \tag{12}$$

which must be false for $m = 1, \ldots, M$ if period $t + \Delta t$ is under concern and thus, when true, indicates an infeasible solution (if there is no initial inventory).

It should be emphasized again, that the construction scheme described above does not necessarily generate an optimum solution. It does not even guarantee to find a feasible solution if there exists one.

4 Randomized Regret Based Sampling

Now, having introduced the backward oriented construction scheme *construct*, a very straightforward idea is to run the construction phase over and over again while memorizing the current best plan until some stopping criterion (e.g. a certain number of iterations) is met. Note, this only makes sense if the *construct*–procedure works non–deterministically.

Here, the choice of setup states j_{mt} comes in again. If a stochastic selection rule is used then it is probable to have different results after each run of the construction phase.

Preliminary studies of ideas reported in this section are provided in [6, 12] for single–level, single–machine PLSP–instances and in [19, 20] for multi–level, single–machine PLSP–instances.

4.1 An Introduction to Random Sampling

The process of random sampling as done here is a Monte Carlo experiment where item numbers j_{mt} are repeatedly drawn at random out of a population $\mathcal{I}_{mt} \subseteq \mathcal{J}_m$

for $m = 1, \ldots, M$ and $t = 1, \ldots, T$. An underlying distribution function φ_{mt} is defined on the basis of a priority value $\pi_{jt} > 0$ that is assigned to each item j in the item set $\mathcal{I}_{mt}$.

Before we give any details of the definitions of φ_{mt} and π_{jt}, let us have a look at a taxonomy of sampling. Depending on the priority values, three important cases are worth to be highlighted. First, there is the general case with a probability to choose $j \in \mathcal{I}_{mt}$ being defined as

$$\varphi_{mt}(j) \overset{def}{=} \frac{\pi_{jt}}{\sum_{i \in \mathcal{I}_{mt}} \pi_{it}} \tag{13}$$

for $m = 1, \ldots, M$ and $t = 1, \ldots, T$ where

$$\sum_{j \in \mathcal{I}_{mt}} \varphi_{mt}(j) = 1 \tag{14}$$

holds. Since the priority values for the items may differ, this is called biased random sampling. By definition, if $j, i \in \mathcal{I}_{mt}$ and $\pi_{jt} > \pi_{it}$, then $\varphi_{mt}(j) > \varphi_{mt}(i)$.

Many authors apply biased random sampling procedures to many different kinds of application areas. A comprehensive overview of research activities is out of the scope of this text. See for instance [8] where a so–called greedy randomized adaptive search procedure (GRASP) is introduced and used for finding maximum independent sets in graphs. Further references to GRASP–applications in the area of corporate acquisition of flexible manufacturing equipment, computer aided process planning, airline flight scheduling and maintenance base planning, and several other problems are given.

As a special case one might choose

$$\pi_{jt} = constant \tag{15}$$

for $j \in \mathcal{I}_{mt}$. This is called pure random sampling,[2] because all items $j \in \mathcal{I}_{mt}$ now have the same probability

$$\varphi_{mt}(j) = \frac{1}{|\mathcal{I}_{mt}|} \tag{16}$$

for $m = 1, \ldots, M$ and $t = 1, \ldots, T$.

Last, since items with a large priority values are preferred, one might compute a modified priority value

$$\varrho_{jt} \overset{def}{=} (\pi_{jt} - \min_{i \in \mathcal{I}_{mt}} \pi_{it} + \epsilon)^{\delta} \tag{17}$$

[2]Note that items $j \in \mathcal{J}_m \backslash \mathcal{I}_{mt}$ have a priority value $\pi_{jt} = 0$. So, we face a pure random sampling only if $\mathcal{I}_{mt}$ is known before the procedure starts, but not if $\mathcal{I}_{mt}$ depends on the history of the execution.

for each $j \in \mathcal{I}_{mt}$ where $\epsilon > 0$ and $\delta \geq 0$, and then define

$$\varphi_{mt}(j) \stackrel{def}{=} \frac{\varrho_{jt}}{\sum_{i \in \mathcal{I}_{mt}} \varrho_{it}} \tag{18}$$

for $m = 1, \ldots, M$ and $t = 1, \ldots, T$. For using such a kind of distribution function (in the context of project scheduling), Drexl coined the name regret based (biased) random sampling [4, 5]. The motivation for this name stems from the idea that the modified priority value ϱ_{jt} represents a measure for the regret not to choose the item j in period t. Hence, more emphasis is given on the differences of priority values π_{jt}. Both, ϵ and δ are method parameters that guide the sampling process.[3] For ϵ a small positive value should be chosen to make sure that every item $j \in \mathcal{I}_{mt}$ is assigned a positive modified priority value $\varrho_{jt} > 0$. The parameter δ amplifies (smoothes) differences in the priority values if $\delta > 1$ $(0 \leq \delta < 1)$.

4.2 Randomized Regret Based Priority Rules

The heart of a random sampling procedure are the priority values π_{jt} that are used to define the distribution function φ_{mt} for $m = 1, \ldots, M$ and $t = 1, \ldots, T$. But before we can introduce priority values, we first need to define the set of items $\mathcal{I}_{mt}$ among which an item j_{mt} is to be chosen.[4] Promising candidates to set a machine m up for at the end of a period t are those items for which there is demand in either period t or, if $t < T$, period $t+1$. In addition to that, items with demand in periods earlier than period t might be promising as well. Choosing such items causes idle periods (during which the setup state can be kept up). Apparently, the cumulative production quantities of an item need not exceed its net requirement. More formally,

$$\begin{aligned} \mathcal{I}_{mt} \stackrel{def}{=} \; & (\{j \in \mathcal{J}_m \mid CD_{j(t+1)} + \tilde{d}_{jt} > 0\} \cup \{j \in \mathcal{J}_m \mid \sum_{\tau=1}^{t-1} \tilde{d}_{j\tau} > 0\}) \\ & \cap \{j \in \mathcal{J}_m \mid nr_j - \sum_{\tau=t+1}^{T} q_{j\tau} > 0\} \end{aligned} \tag{19}$$

for $m = 1, \ldots, M$ and $t = 1, \ldots, T$. If $\mathcal{I}_{mt} = \emptyset$, we simply choose $j_{mt} = j_{m(t+1)}$ in periods $t < T$, or, if $t = T$, fix $j_{mT} \in \mathcal{J}_m$ by arbitration, e.g. the item with the lowest item index (which when turned out to be wrong is neatly be corrected by the postprocessor). Let us therefore assume $\mathcal{I}_{mt} \neq \emptyset$.

Using our problem understanding, two main aspects help us to find priority values. On the one hand, we like to have low-cost production plans. A priority

[3]Using different parameters ϵ_{mt} and δ_{mt} for each machine m and each period t turned out to increase the run-time, but did not improve the results decidedly.

[4]Remember, in some cases which are described in Section 3 no selection is to be made, because we do not allow lot splitting.

value should therefore reflect cost criteria. On the other hand, it is quite hard to generate even a feasible solution. Thus, priority values should also consider sources of infeasibility. In combination, priority values should lead to cheap and feasible production plans.

We start with introducing two expressions that represent cost criteria. First, imagine that a machine m is not set up for an item j at the end of period t. This means that $CD_{j(t+1)}$ items must be stored in inventory for at least one additional period and thus causing additional holding costs. The expression

$$\pi_{jt}^{I} \overset{def}{=} \frac{h_j CD_{j(t+1)}}{\max\{s_i \mid i \in \mathcal{J}_m\}} \tag{20}$$

for $m = 1, \ldots, M$, $t = 1, \ldots, T$, and $j \in \mathcal{I}_{mt}$ is a measure for these costs.

Second, changing the setup state causes setup costs, or, stating this the other way around, not to select a certain item may save setup costs. Thus,

$$\pi_{jt}^{II} \overset{def}{=} \frac{s_j}{\max\{s_i \mid i \in \mathcal{J}_m\}} \tag{21}$$

for $m = 1, \ldots, M$, $t = 1, \ldots, T$, and $j \in \mathcal{I}_{mt}$ is a measure for the cost savings, if an item is not selected.

In addition to that, we now give two expressions that tend to avoid infeasibility. On the one hand, we take into account that items usually have preceding items which are to be manufactured in advance. The more preceding items there are, the more risky it is to shift production into early periods. The closer we move towards period 1 the more important is this aspect. Using

$$dep_j \overset{def}{=} \begin{cases} 0 & \text{, if } \mathcal{P}_j = \emptyset \\ \max\limits_{i \in \mathcal{P}_j}\{v_i + dep_i\} & \text{, otherwise} \end{cases} \quad j = 1, \ldots, J \tag{22}$$

to denote the depth of an item j, the expression

$$\pi_{jt}^{III} \overset{def}{=} \frac{dep_j}{t + 1 - dep_j} \tag{23}$$

for $m = 1, \ldots, M$, $t = 1, \ldots, T$, and $j \in \mathcal{I}_{mt}$ thus makes it more probable to choose items with a large depth, especially in early periods. To be well-defined choose a denominator equal to one if $t + 1 = dep_j$.

On the other hand, we consider the capacity usage. Producing an item j and all its preceding items requires a well-defined amount of capacity per machine. The bottleneck machine can then be defined as the machine with the highest ratio of capacity demand per available capacity. Items which cause a high capacity usage on the bottleneck machine should therefore have a high preference to be chosen. More formally,

$$\begin{aligned} \pi_{jt}^{IV} \overset{def}{=} & \; (CD_{j(t+1)} + \tilde{d}_{jt}) \\ & \cdot \max\left\{ \frac{\sum_{i \in (\mathcal{P}_j \cup \{j\}) \cap \mathcal{J}_m} p_i i d_{ji}}{\sum_{\tau=1}^{t} C_{m\tau}} \mid m \in \{1, \ldots, M\} \right\} \end{aligned} \tag{24}$$

for $m = 1, \ldots, M$, $t = 1, \ldots, T$, and $j \in \mathcal{I}_{mt}$ estimates the capacity usage of the bottleneck machine.

As preliminary tests have revealed, using a combination of these four criteria seems to be a good strategy. For $m \in \{1, \ldots, M\}$, $t \in \{1, \ldots, T\}$, and $j \in \mathcal{I}_{mt}$ four cases may now occur where $j_{m(T+1)} = 0$ is assumed for notational convenience:

Case 1: $CD_{j(t+1)} + \tilde{d}_{jt} > 0$ and $j \neq j_{m(t+1)}$.
Case 2: $CD_{j(t+1)} + \tilde{d}_{jt} > 0$ and $j = j_{m(t+1)}$.
Case 3: $\sum_{\tau=1}^{t-1} \tilde{d}_{j\tau} > 0$ and $j \neq j_{m(t+1)}$ and Case 1 does not hold.
Case 4: $\sum_{\tau=1}^{t-1} \tilde{d}_{j\tau} > 0$ and $j = j_{m(t+1)}$ and Case 2 does not hold.

Depending on the case that holds, π_{jt} can now be defined as

$$\pi_{jt} \stackrel{def}{=} \begin{cases} \gamma_1 \pi_{jt}^{I} - \gamma_2 \pi_{jt}^{II} + \gamma_3 \pi_{jt}^{III} + \gamma_4 \pi_{jt}^{IV} & \textit{Case 1} \\ \gamma_1 \pi_{jt}^{I} + \gamma_3 \pi_{jt}^{III} + \gamma_4 \pi_{jt}^{IV} & \textit{Case 2} \\ \gamma_3 \pi_{jt}^{III} + \gamma_4 \pi_{jt}^{IV} & \textit{Case 3} \\ \gamma_2 \pi_{jt}^{II} + \gamma_3 \pi_{jt}^{III} + \gamma_4 \pi_{jt}^{IV} & \textit{Case 4} \end{cases} \tag{25}$$

where $\gamma_1, \ldots, \gamma_4$ are real–valued method parameters.[5] Cases 1 and 2 are defined as motivated above. Note, that in Case 2 the setup cost savings are of no interest, because the machine already is in the proper setup state. In the Cases 3 and 4, the holding cost criterion evaluates to zero. In contrast to the first two cases, the setup cost criterion now increases the priority value if the machine is in the correct setup state (Case 4) and is neglected in Case 3. This differs from what is done in Cases 1 and 2, although its interpretation, i.e. giving items for which the machine is already set up for a better chance to be selected, is the same. The reason for doing so is motivated by a desire to decide for idle periods in order to maintain the setup state (Case 4). This idea is supported by making the priority value for the item for which Case 4 holds large and lower the priority values for those items which apply to Case 1.

Using (18) in combination with (17) we now have a full specification for the selection rule we employ. If the method parameters are chosen at random, the solution procedure is called randomized regret based (biased random) sampling.

4.3 Tuning the Method Parameters

In our implementation we choose all method parameters, namely $\gamma_1, \ldots, \gamma_4$, ϵ, and δ, at random in each iteration of the sampling process. Rather than doing this at pure random over and over again without any history sensitivity, one should make use of the information gained during previous iterations. This is to say that learning effects should steer the choice of the method parameters for values that tend to give good results.

[5]Using different parameters $\gamma_{1mt}, \ldots, \gamma_{4mt}$ for each machine m and each period t turned out to increase the computational overhead without giving decidedly better results.

Since all that will be said is valid for all parameters, let

$$parameter \in \{\gamma_1, \ldots, \gamma_4, \epsilon, \delta\} \tag{26}$$

denote any of the method parameters. Furthermore, suppose $n = 1, 2, \ldots$ is the number of the current iteration and $parameter^*$ is the value of the parameter when the current best (feasible) solution was found within the last $n - 1$ iterations. Initially, $parameter^* = 0$ without loss of generality. In each iteration we randomly choose a parameter value, say pv, out of the interval of valid parameter values, say $[parameter_{min}, parameter_{max}]$, as specified by the user. If we would assign $parameter = pv$, we have a pure random choice without any learning.

A fundamental idea for tuning the method parameters is to intensify the search in those areas of the parameter space which are in the neighborhood of $parameter^*$. In our implementation, this is done using

$$parameter = parameter^* + \nu_n(pv - parameter^*) \tag{27}$$

where $\nu_n \in [0, 1]$ defines the neighborhood around the current best parameter value from which the parameter used in iteration n is to be selected. If $\nu_n = 1$ then we have a pure random choice of parameters again. And if $\nu_n = 0$ then we would have no choice at all, because in every iteration $parameter^*$ would be chosen again. Initially, we start with $\nu_1 = 1$. Note that in each iteration the parameter value lies indeed in the valid parameter space, i.e.

$$parameter_{min} \leq parameter \leq parameter_{max}. \tag{28}$$

Without any significant effort, we can count the number of iterations done so far that resulted in improved solutions. Let this number be denoted as *improvements*. Starting with a counter $improvements = 0$, its value is set to one if the first feasible solution is found, and its value is increased by one every time a new current best solution is found. Having this information, we compute

$$\nu_{n+1} = \nu_n \tag{29}$$

if iteration n did not improve the current best solution and

$$\nu_{n+1} = \frac{1}{improvements} \tag{30}$$

if iteration n resulted in an improvement. Note, because the counter *improvements* monotonically increases from iteration to iteration, the value of ν_n declines during run–time and intensifies the search in a more and more narrow subspace of the parameter space.

In preliminary experiments we found out, that the intensification procedure described so far quite often traps into local optima, because the value ν_n declines too fast. The final outcome is often worse than what comes out with a pure random choice of parameter values in every iteration. Thus, we decided to choose $\nu_{n+1} = \nu_n$ for $n \leq NOINTENSIFY$ where $NOINTENSIFY$ is specified by the user.[6] Nevertheless, the counter *improvements* is maintained during the first $NOINTENSIFY$ iterations as well.

In addition to that, we found out that for some instances it is quite hard to find a feasible solution. In such cases, intensification turned out to be good. For instances for which feasible solutions are easily found intensification is less exciting. A counter for the number of iterations which revealed no feasible solution, namely $infeasible$, thus allows to decrease the parameter ν_n only if

$$\frac{infeasible}{n} > CRITICAL \tag{31}$$

where $CRITICAL \in [0,1]$ is a user–specified parameter.

4.4 Modifications of the Construction Scheme

The construction scheme given in Section 3 would work fine without any modification. However, suppose that we have chosen to set machine m up for item $j_{mt} \in \mathcal{I}_{mt}$ at the end of period t. Furthermore, assume that

$$j_{mt} \notin \{j \in \mathcal{J}_m \mid CD_{j(t+1)} + \tilde{d}_{jt} > 0\} \tag{32}$$

which implies that

$$j_{mt} \in \{j \in \mathcal{J}_m \mid \sum_{\tau=1}^{t-1} \tilde{d}_{j\tau} > 0\}. \tag{33}$$

Let $t_{j_{mt}}$ be the next period in which demand for item j_{mt} occurs, i.e.

$$t_{j_{mt}} = \max\{\tau \mid 1 \leq \tau < t \wedge \tilde{d}_{j_{mt}\tau} > 0\}. \tag{34}$$

Since the construction procedure strictly moves on from period to period choosing items again and again, it is quite unlikely to begin in period $t > t_{j_{mt}}$ and reach period $t_{j_{mt}}$ having no other item than j_{mt} be chosen for machine m at the end of periods $t_{j_{mt}}, t_{j_{mt}} + 1, \ldots, t$.

For this reason, we introduce the following modification: If an item j_{mt} fulfilling (32) is chosen then we update the $\tilde{d}_{jt}$–entries as described in Table 7.

This makes it unlikely to choose any other item than j_{mt} to be produced on machine m in the periods $t_{j_{mt}} + 1, \ldots, t-1$, because all demand that would require machine m is shifted to period $t_{j_{mt}}$. Note, there is no guarantee to choose j_{mt} again and again. This makes sense, because after j_{mt} is selected at

[6]For instance, during tests with $J = 5$ and $T = 10$ we observed, that after 500 iterations we very often have results that are close to the results after 1,000 iterations when a pure random choice of parameter values is done [20]. So, we would suggest $NOINTENSIFY = 500$ for these instances.

```
for $j \in \mathcal{J}_m$
    $\tilde{d}_{jt_{j_{mt}}} := CD_{j(t+1)} + \sum_{\tau=t_{j_{mt}}}^{t} \tilde{d}_{j\tau}$.
    $CD_{j(t+1)} := 0$.
    for $\tau \in \{t_{j_{mt}} + 1, \ldots, t\}$
        $\tilde{d}_{j\tau} := 0$.
```

Table 7: Generating Idle Periods $t_{j_{mt}} + 1, \ldots, t$

the end of period t it may happen that scheduling items on machines other than machine m causes updates of the $\tilde{d}_{j\tau}$-entries where $t_{j_{mt}} < \tau < t$. This demand has previously not been taken into account and hence we decide again if the setup state of machine m should indeed be kept up for item j_{mt}.

5 Experimental Evaluation

To test the performance of the randomized regret based sampling method we use the test–bed defined in [21] which is a collection of 1080 instances with $J = 5$ and $T = 10$. This set of instances is defined by a full factorial experimental design. The problem parameters of interest are:

- $M \in \{1, 2\}$, the number of machines.
- $C \in \{0.2, 08\}$, the complexity of the gozinto–structure. Roughly speaking, a complexity close to 1 means many arcs in the gozinto–structure, and a complexity close to 0 stands for few arcs (see [21] for a formal definition).
- $(T_{macro}, T_{micro}, T_{idle}) \in \{(10, 1, 5), (5, 2, 2), (1, 10, 0)\}$, the demand pattern. The planning horizon is subdivided into T_{macro} macro periods each of which consists of T_{micro} micro periods. External demand occurs at the end of each macro period out of the first T_{idle} macro periods. $(5, 2, 2)$ thus means that we face external demand in periods 6, 8, and 10.
- $COSTRATIO \in \{5, 150, 900\}$, the average ratio of setup and holding costs per item.
- $U \in \{30, 50, 70\}$, the percentage of capacity utilization.

For each parameter level combination we generated 10 instances using the idea of common random numbers which gives a total of 1080 instances. 1033 of them turned out to have a feasible solution. In each iteration the method parameters are drawn at random so that $\gamma_1, \ldots, \gamma_4 \in [0, 1]$, $\epsilon \in [0.0001, 0.1]$, and $\delta \in [0, 10]$. For tuning the method parameters we use $NOINTENSIFY = 500$

and $CRITICAL = 0.6$. For each parameter level combination we find the average deviation of the upper bound from the optimum solution where for each instance the deviation is computed by

$$deviation \overset{def}{=} 100\frac{UB - OPT}{OPT} \tag{35}$$

with UB being the upper bound and OPT the optimum objective function value. Moreover, the tables show the worst case deviation and the number of instances for which a feasible solution is found, too.

An analysis shall now be done on the basis of aggregated data to see whether or not certain parameter levels have an impact on the performance of the solution procedure. For each parameter level we give three performance measures. The average deviation from the optimum results is, perhaps, the most important. But, the infeasibility ratio which is the percentage of instances for which no feasible solution can be found although such one exists is a good indicator of what makes instances hard to solve, too. The average run–time performance is also shown where all values are given in CPU–seconds on a Pentium computer with 120 MHz.

The effect of changing the number of machines is studied in Table 8. As we see, both, the average deviation from optimum and the average run–time significantly increase with the number of machines. However, more machines make finding a feasible solution easier.

	$M = 1$	$M = 2$
Average Deviation	8.90	11.69
Infeasibility Ratio	11.46	7.92
Average Run–Time	0.39	0.51

Table 8: The Impact of the Number of Machines on the Performance

Varying the gozinto–structure complexity also effects the performance of the regret based solution procedure. Table 9 provides the details. A high complexity results in larger deviations from the optimum than a low complexity. It also raises the infeasibility ratio decidedly. A minor impact is on the run–time performance where gozinto–stuctures with a greater complexity have a small tendency to increase the computational effort very slightly on average.

Table 10 shows if the demand pattern affects the performance of the heuristic, too. We learn that a pattern with many non–zeroes in the demand matrix gives poor results. The average deviation from the optimum is positively correlated with the number of demands to be fulfilled. For sparsely–filled demand matrices the infeasibility ratio is below 1%, but increases dramatically for demand matrices with many entries. Differences in the run–time performance are not worth to be mentioned.

	$C = 0.2$	$C = 0.8$
Average Deviation	9.23	11.53
Infeasibility Ratio	7.05	12.40
Average Run–Time	0.44	0.47

Table 9: The Impact of the Gozinto–Structure Complexity on the Performance

	$(T_{macro}, T_{micro}, T_{idle}) =$		
	$(10, 1, 5)$	$(5, 2, 2)$	$(1, 10, 0)$
Average Deviation	16.25	12.63	3.52
Infeasibility Ratio	14.53	14.20	0.84
Average Run–Time	0.48	0.44	0.44

Table 10: The Impact of the Demand Pattern on the Performance

The impact of the cost structure on the performance is analyzed in Table 11. It can be seen that for high setup costs the deviation of the upper bound from the optimum objective function value is on average significantly smaller than for low setup costs. The infeasibility ratios show that instances with very high or very low setup costs are not as easy to solve as instances with a balanced cost structure. The influence of different costs on the run–time performance can be neglected.

	$COSTRATIO =$		
	5	150	900
Average Deviation	14.42	8.95	7.65
Infeasibility Ratio	10.72	8.12	10.20
Average Run–Time	0.46	0.45	0.44

Table 11: The Impact of the Cost Structure on the Performance

The results for different capacity utilizations are provided in Table 12. The average deviation from the optimum is positively correlated with the capacity utilization. Differences are, however, not dramatic. Major effects are for the infeasibility ratios. While for each instance with a low capacity utilization a feasible solution can be found, one out of three instances with a capacity utilization $U = 70$ cannot be solved. The run–time performance is almost unaffected by changes in the capacity utilization.

	$U = 30$	$U = 50$	$U = 70$
Average Deviation	9.10	10.99	11.24
Infeasibility Ratio	0.00	2.23	28.84
Average Run-Time	0.46	0.46	0.44

Table 12: The Impact of the Capacity Utilization on the Performance

In summary, we find that 100 out of the 1,033 instances in our test-bed cannot be solved by the randomized regret based sampling method. This gives an overall infeasibility ratio of 9.68%. The average run-time is 0.45 CPU-seconds. The overall average deviation of the upper bound from the optimum objective function value is 10.33%.

The most significant method parameter certainly is the number of iterations that are to be performed. Hence, Table 13 gives some insight into performance changes due to that value. As a point of reference we also give the results for a variant of the randomized regret based sampling procedures which stops when a first feasible solution is found or when 1,000 iterations are performed.

	Average Deviation	Infeasibility Ratio	Average Run-Time
First Solution	31.93	9.68	0.02
100 Iterations	13.39	18.01	0.05
500 Iterations	11.00	12.58	0.23
1,000 Iterations	10.33	9.68	0.45
2,000 Iterations	9.78	8.23	0.90

Table 13: The Impact of the Number of Iterations on the Performance

We see that performing 1,000 iterations is a good choice, because even if we would double that value the average deviation from the optimum objective function values would not be drastically reduced. The poor average deviation results for the first feasible solutions that are found indicates that the instances in the test-bed are indeed not easy to solve. Hence, we can state that the randomized regret based sampling procedure indeed makes a contribution.

6 Parallel Machines

Lot sizing and scheduling with parallel machines (PLSP-PM) introduces a new degree of freedom into the planning problem. In contrast to lot sizing and scheduling with multiple machines we now have no fixed machine assignments for the items. In other words, it is a priori not clear on which machine items are produced. We assume that some (maybe all) machines are capable to produce a particular item. The multi–machine case obviously is a special case of the parallel–machine case, because the number of machines to choose among is one for each item. This actually means that there is no choice. Furthermore, we assume the most general case that is the case of heterogeneous machines. This is to say, that the production of the same item requires different amounts of capacity if different machines are used.

Table 14 defines the decision variables which are new or redefined. Likewise, Table 15 gives the parameters. A MIP–model formulation can now be presented to give a precise problem statement.

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

Table 14: New Decision Variables for the PLSP–PM

Symbol	Definition
$\mathcal{J}_m$	Set of all items that share the machine m, i.e. $\mathcal{J}_m \overset{def}{=} \{j \in \{1, \ldots, J\} \mid m \in \mathcal{M}_j\}$.
$\mathcal{M}_j$	Set of all machines that are capable to produce item j, i.e. $\mathcal{M}_j \overset{def}{=} \{m \in \{1, \ldots, M\} \mid p_{jm} < \infty\}$.
p_{jm}	Capacity needs for producing one unit of item j on machine m. Its value is ∞ if machine m cannot be used to produce item j.
s_{jm}	Non–negative setup cost for item j on machine m.
y_{jm0}	Initial setup state.

Table 15: New Parameters for the PLSP–PM

$$\min \sum_{m=1}^{M} \sum_{j \in \mathcal{J}_m} \sum_{t=1}^{T} s_{jm} x_{jmt} \tag{36}$$

$$+ \sum_{j=1}^{J} \sum_{t=1}^{T} h_j I_{jt}$$

subject to

$$I_{jt} = I_{j(t-1)} + \sum_{m \in \mathcal{M}_j} q_{jmt} - d_{jt} - \sum_{i \in \mathcal{S}_j} \sum_{m \in \mathcal{M}_i} a_{ji} q_{imt} \qquad j = 1, \ldots, J, \; t = 1, \ldots, T \tag{37}$$

$$I_{jt} \geq \sum_{i \in \mathcal{S}_j} \sum_{m \in \mathcal{M}_i} \sum_{\tau = t+1}^{\min\{t+v_j, T\}} a_{ji} q_{im\tau} \qquad j = 1, \ldots, J, \; t = 0, \ldots, T-1 \tag{38}$$

$$\sum_{j \in \mathcal{J}_m} y_{jmt} \leq 1 \qquad m = 1, \ldots, M, \; t = 1, \ldots, T \tag{39}$$

$$x_{jmt} \geq y_{jmt} - y_{jm(t-1)} \qquad j = 1, \ldots, J, \; m \in \mathcal{M}_j, \; t = 1, \ldots, T \tag{40}$$

$$p_{jm} q_{jmt} \leq C_{mt}(y_{jm(t-1)} + y_{jmt}) \qquad j = 1, \ldots, J, \; m \in \mathcal{M}_j, \; t = 1, \ldots, T \tag{41}$$

$$\sum_{j \in \mathcal{J}_m} p_{jm} q_{jmt} \leq C_{mt} \qquad m = 1, \ldots, M, \; t = 1, \ldots, T \tag{42}$$

$$y_{jmt} \in \{0, 1\} \qquad j = 1, \ldots, J, \; m \in \mathcal{M}_j, \; t = 1, \ldots, T \tag{43}$$

$$I_{jt} \geq 0 \qquad j = 1, \ldots, J, \; t = 1, \ldots, T \tag{44}$$

$$q_{jmt}, x_{jmt} \geq 0 \qquad j = 1, \ldots, J, \; m \in \mathcal{M}_j, \; t = 1, \ldots, T \tag{45}$$

The meaning of (36) to (45) closely relates to (1) to (9) and thus needs no further explanation. However, it is remarkable to note that lots of the same item may be produced on different machines in the same period. Splitting lots for concurrent production on different machines may be necessary to find feasible (and optimum) solutions.

7 Multiple Resources

For lot sizing and scheduling with multiple resources (*PLSP–MR*) we assume that each item requires several resources at a time. Manufacturing an item needs all corresponding resources to be in the right setup state. Thus, the model has to guarantee that, if an item is produced in a period, all required resources are in the proper setup state either at the beginning or at the end of the period. The multi-machine problem is a special case of multiple resources. In the former case items require only one resource to be produced.

Table 16 introduces the new decision variables and Table 17 gives the new parameters that are needed to extend the multi-machine problem. A MIP-model formulation using this notation can now be presented.

Symbol	Definition
q_{jt}^B	Production quantity for item j in period t where the required resources are properly set up at the beginning of period t.
q_{jt}^E	Production quantity for item j in period t where the required resources are properly set up at the end of period t.
x_{jmt}	Binary variable which indicates whether a setup for item j occurs on resource m in period t ($x_{jmt} = 1$) or not ($x_{jmt} = 0$).
y_{jmt}	Binary variable which indicates whether resource m is set up for item j at the end of period t ($y_{jmt} = 1$) or not ($y_{jmt} = 0$).

Table 16: New Decision Variables for the PLSP–MR

$$\min \sum_{m=1}^{M} \sum_{j \in \mathcal{J}_m} \sum_{t=1}^{T} s_{jm} x_{jmt} + \sum_{j=1}^{J} \sum_{t=1}^{T} h_j I_{jt} \tag{46}$$

subject to

$$I_{jt} = I_{j(t-1)} + q_{jt}^B + q_{jt}^E - d_{jt} - \sum_{i \in \mathcal{S}_j} a_{ji}(q_{it}^B + q_{it}^E) \qquad j = 1, \ldots, J;\ t = 1, \ldots, T \tag{47}$$

Symbol	Definition
$\mathcal{J}_m$	Set of all items that share the resource m, i.e. $\mathcal{J}_m \overset{def}{=} \{j \in \{1,\ldots,J\} \mid p_{jm} < \infty\}$.
$\mathcal{M}_j$	Set of all resources that are needed to produce item j, i.e. $\mathcal{M}_j \overset{def}{=} \{m \in \{1,\ldots,M\} \mid p_{jm} < \infty\}$.
p_{jm}	Capacity needs for resource m for producing one unit of item j. Its value is ∞ if item j does not require resource m.
s_{jm}	Non-negative setup cost for item j on resource m.
y_{jm0}	Initial setup state.

Table 17: New Parameters for the PLSP-MR

$$I_{jt} \geq \sum_{i \in \mathcal{S}_j} \sum_{\tau=t+1}^{\min\{t+v_j,T\}} a_{ji}(q_{i\tau}^B + q_{i\tau}^E) \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 0,\ldots,T-1 \end{array} \tag{48}$$

$$\sum_{j \in \mathcal{J}_m} y_{jmt} \leq 1 \qquad \begin{array}{l} m = 1,\ldots,M \\ t = 1,\ldots,T \end{array} \tag{49}$$

$$x_{jmt} \geq y_{jmt} - y_{jm(t-1)} \qquad \begin{array}{l} j = 1,\ldots,J \\ m \in \mathcal{M}_j \\ t = 1,\ldots,T \end{array} \tag{50}$$

$$p_{jm} q_{jt}^B \leq C_{mt} y_{jm(t-1)} \qquad \begin{array}{l} j = 1,\ldots,J \\ m \in \mathcal{M}_j \\ t = 1,\ldots,T \end{array} \tag{51}$$

$$p_{jm} q_{jt}^E \leq C_{mt} y_{jmt} \qquad \begin{array}{l} j = 1,\ldots,J \\ m \in \mathcal{M}_j \\ t = 1,\ldots,T \end{array} \tag{52}$$

$$\sum_{j \in \mathcal{J}_m} p_{jm}(q_{jt}^B + q_{jt}^E) \leq C_{mt} \qquad \begin{array}{l} m = 1,\ldots,M \\ t = 1,\ldots,T \end{array} \tag{53}$$

$$y_{jmt} \in \{0,1\} \qquad \begin{array}{l} j = 1,\ldots,J \\ m \in \mathcal{M}_j \\ t = 1,\ldots,T \end{array} \tag{54}$$

$$I_{jt}, q_{jt}^B, q_{jt}^E \geq 0 \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 1,\ldots,T \end{array} \tag{55}$$

$$x_{jmt} \geq 0 \qquad \begin{array}{l} j = 1,\ldots,J \\ m \in \mathcal{M}_j \\ t = 1,\ldots,T \end{array} \tag{56}$$

This MIP–formulation is a straightforward extension of the PLSP–MM–model and thus needs no elaborate discussion. However, an important aspect should be mentioned briefly. Due to the decision variables q_{jt}^B and q_{jt}^E we explicitly take into account the sequence of two different lots in a period. This is redundant in the PLSP–MM–model since the setup state variables y_{jmt} uniquely define sequences. But now, we must make sure that all resources needed to produce an item j are in the right setup state at a time. Constraints (51) and (52) guarantee so. Noteworthy to say that postprocessing as described above cannot be done here since the schedules of different resources do interact.

8 Partially Renewable Resources

For lot sizing and scheduling with partially renewable resources (PLSP–PRR) we assume that each item requires one resource for which a setup state has to be taken into account. This is equivalent to the multi–machine case. In addition, each item may require several scarce resources for which no setup state has to be taken into account. Moreover, since we have small periods representing shifts or hours for instance, we assume that these additional resources have capacity limits given per interval of periods.[7] Suppose for example, that periods represent shifts (e.g. 10 shifts per week) and resources are renewed once a week. Since they are not renewed in every period they are called partially renewable. Capacity limits are then given per week and not per shift. The case where capacity limits are given per period is of course a special case. Such resources would be called renewable. The one for which a setup state is taken into account is a renewable resource.

To understand the subsequent MIP–model formulation, see Table 18 for new parameters. The decision variables equal those of the PLSP–MM–model.

$$\min \sum_{j=1}^{J} \sum_{t=1}^{T} (s_j x_{jt} + h_j I_{jt}) \tag{57}$$

subject to

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt} - \sum_{i \in \mathcal{S}_j} a_{ji} q_{it} \qquad \begin{array}{l} j = 1, \ldots, J \\ t = 1, \ldots, T \end{array} \tag{58}$$

$$I_{jt} \geq \sum_{i \in \mathcal{S}_j} \sum_{\tau = t+1}^{\min\{t+v_j, T\}} a_{ji} q_{i\tau} \qquad \begin{array}{l} j = 1, \ldots, J \\ t = 0, \ldots, T-1 \end{array} \tag{59}$$

$$\sum_{j \in \mathcal{J}_m} y_{jt} \leq 1 \qquad \begin{array}{l} m = 1, \ldots, M \\ t = 1, \ldots, T \end{array} \tag{60}$$

[7] A more general point of view would be to consider arbitrary sets of periods instead of intervals of periods. However, this is not done here.

Symbol	Definition
$\tilde{C}_{mt}$	Available capacity of the partially renewable resource m in interval t.
$\tilde{\mathcal{J}}_m$	Set of all items that share the partially renewable resource m, i.e. $\tilde{\mathcal{J}}_m \overset{def}{=} \{j \in \{1,\ldots,J\} \mid \tilde{p}_{jm} < \infty\}$.
$\tilde{L}$	Length of an interval in number of periods. All intervals are assumed to have equal length.
$\tilde{M}$	Number of partially renewable resources.
$\tilde{\mathcal{M}}_j$	Set of all partially renewable resources that are needed to produce item j, i.e. $\tilde{\mathcal{M}}_j \overset{def}{=} \left\{m \in \{1,\ldots,\tilde{M}\} \mid \tilde{p}_{jm} < \infty\right\}$.
$\tilde{p}_{jm}$	Capacity needs for the partially renewable resource m for producing one unit of item j. Its value is ∞ if item j does not require resource m.
$\tilde{T}$	Number of intervals. We assume $\tilde{L} \cdot \tilde{T} = T$ and $[(i-1)\tilde{L}+1, i\tilde{L}]$ be the i–th interval of periods where $i = 1,\ldots,\tilde{T}$. Note, a period $t \in \{1,\ldots,T\}$ belongs to the interval $i = \left\lceil \frac{t}{\tilde{L}} \right\rceil$.

Table 18: New Parameters for the PLSP–PRR

$$x_{jt} \geq y_{jt} - y_{j(t-1)} \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 1,\ldots,T \end{array} \tag{61}$$

$$p_j q_{jt} \leq C_{m_j t}(y_{j(t-1)} + y_{jt}) \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 1,\ldots,T \end{array} \tag{62}$$

$$\sum_{j \in \mathcal{J}_m} p_j q_{jt} \leq C_{mt} \qquad \begin{array}{l} m = 1,\ldots,M \\ t = 1,\ldots,T \end{array} \tag{63}$$

$$\sum_{j \in \tilde{\mathcal{J}}_m} \sum_{\tau=(t-1)\tilde{L}+1}^{t\tilde{L}} \tilde{p}_{jm} q_{j\tau} \leq \tilde{C}_{mt} \qquad \begin{array}{l} m = 1,\ldots,\tilde{M} \\ t = 1,\ldots,\tilde{T} \end{array} \tag{64}$$

$$y_{jt} \in \{0,1\} \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 1,\ldots,T \end{array} \tag{65}$$

$$I_{jt}, q_{jt}, x_{jt} \geq 0 \qquad \begin{array}{l} j = 1,\ldots,J \\ t = 1,\ldots,T \end{array} \tag{66}$$

All constraints but (64) equal those of the PLSP–MM–model and thus need no explanation again. The new set of restrictions (64) represents the capacity limits of the partially renewable resources. They make sure that the sum of capacity demands for resource m in the t–th interval of periods does not exceed the capacity limit.

9 Conclusion

In this paper we tackled the multi–level proportional lot sizing and scheduling problem. Starting with a mixed–integer model formulation for multiple machines where each item requires exactly one machine, we presented a rather general idea of constructing production plans. By refining this generic construction scheme, we introduced a randomized regret based sampling method which can be seen as a generalization of the BACKADD–procedure given in [6]. A computational study revealed that even for the thorny problem of multi–level lot sizing and scheduling with multiple machines, feasible solutions with an average deviation of about 10% from the optimum objective function value can be computed within less than half a second CPU–time on a Pentium P120 computer. Other mixed–integer models for parallel machines, multiple resources, and partially renewable resources are also given. In summary, we find the proportional lot sizing and scheduling problem worth to be considered. Future work should implement variants of the construction scheme for the extension models and test their performance.

Acknowledgement

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

References

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

[2] 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

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

[4] DREXL, A., (1991), Scheduling of Project Networks by Job Assignment, Management Science, Vol. 37, pp. 1590–1602

[5] DREXL, A., GRÜNEWALD, J., (1993), Nonpreemptive Muli–Mode Resource–Constrained Project Scheduling, IIE Transactions, Vol. 25, No. 5, pp. 74–81

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

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

[8] FEO, T.A., RESENDE, M.G.C., SMITH, S.H., (1994), A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set, Operations Research, Vol. 42, pp. 860–878

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

[10] 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

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

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

[13] 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

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

[15] 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

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

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

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

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

[20] 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

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

[22] 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

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

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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