Sahling, Florian; Buschkühl, Lisbeth; Tempelmeier, Horst; Helber, Stefan

Working Paper

Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic

Diskussionsbeitrag, No. 400

Provided in Cooperation with:
School of Economics and Management, University of Hannover

Solving a Multi-Level Capacitated Lot Sizing Problem with Multi-Period Setup Carry-Over via a Fix-and-Optimize Heuristic

Florian Sahling[a], Lisbeth Buschkühl[b],

Horst Tempelmeier[b] and Stefan Helber[a]

[a]Department of

Production Management

Leibniz University Hannover, Germany

[b]Department of

Supply Chain Management and Production

University of Cologne, Germany

June 2, 2008

Abstract

This paper presents a new algorithm for the dynamic Multi-Level Capacitated Lot Sizing Problem with Setup Carry-Overs (MLCLSP-L). The MLCLSP-L is a big-bucket model that allows the production of any number of products within a period, but it incorporates partial sequencing of the production orders in the sense that the first and the last product produced in a period are determined by the model. We solve a model which is applicable to general bill-of-material structures and which includes minimum lead times of one period and multi-period setup carry-overs. Our algorithm solves

a series of mixed-integer linear programs in an iterative so-called Fix-and-Optimize approach. In each instance of these mixed-integer linear programs a large number of binary setup variables is fixed whereas only a small subset of these variables is optimized, together with the complete set of the inventory and lot size variables. A numerical study shows that the algorithm provides high-quality results and that the computational effort is moderate.

1 Introduction

Capacitated dynamic lot sizing deals with the problem of determining time-phased production quantities that meet both given external demands and given capacity limits of the production system. The problem arises in production environments where the changeover of a resource from one product type to another causes setup time and/or setup costs. Numerous models to support dynamic lot sizing for single-level as well as multi-level production systems have been presented over the last decades.

Big-bucket model formulations such as the (single-level) Capacitated Lot Sizing Problem (CLSP) and its multi-level extension, the Multi-Level Capacitated Lot Sizing Problem (MLCLSP), determine production quantities and periods only, without consideration of the actual production sequence of the orders within a time period. This type of modeling has the advantage that it allows a flexible re-sequencing of orders within a period, at the cost, however, that a detailed production plan must be generated in a subsequent planning step. On the other hand, a number of model variations completely integrate lot sizing and sequencing decisions, however, at a significant computational cost. These so-called small-bucket models are known as the Discrete Lot Sizing and Scheduling Problem (DLSP), the Continuous Setup Lot Sizing Problem (CSLP) and the Proportional Lot Sizing and Scheduling Problem (PLSP), among others. For recent detailed reviews on these models see e. g. Staggemeier and Clark (2001) and Sürie and Stadtler (2003).

In order to ensure that an MLCLSP solution can be transformed into a feasible

production schedule, a planned lead time of at least one period must be introduced for each component product. In a multi-level bill-of-material (BOM) structure the cumulated flow time (in periods) from the beginning of the processing of the raw material to the completion of the finished product is then equal to the number of levels in the BOM structure. As for a period length of one week with a ten-level BOM structure this would result in a minimum flow time of at least ten weeks, the cumulated flow time can only be shortened by a reduction of the period length to, say, one day. However, if the time buckets are too small, then only a small number of products will be produced within a single period and in this case it will often happen that production in period t is continued in period $t+1$ (and possibly periods $t+2, \ldots$) without an additional setup. The big-bucket MLCLSP formulation counts this as a second setup, and therefore with short period lengths provides only a rough estimate of the real number of setups.

The Multi-Level Capacitated Lot Sizing Problem with Linked Lot Sizes (MLCLSP-L), which is an extension of the big-bucket MLCLSP, allows to carry over the setup state of a resource to the next periods following the setup. This leads to more efficient setup patterns and shorter planning-induced flow times. In addition, due to its more realistic book-keeping of setups, it allows to find a feasible solution in cases when the standard MLCLSP formulation would fail. This is particularly likely in situations with high utilization.

In the current paper, we consider the MLCLSP-L with multiple setup carry-overs that allows to preserve the setup state of a resource over multiple periods. We apply the Fix-and-Optimize heuristic presented in Helber and Sahling (2008) to this variant of the MLCLSP-L. The Fix-and-Optimize heuristic, which is directly based on the formulation of the problem as a linear mixed-integer program, is rather flexible with respect to the incorporation of additional constraints. For example, constraints such as minimum lot sizes or maximum inventory levels can easily be included into the model without requiring a change to the solution approach. In addition, it has shown to outperform other well-known heuristics for multi-level lot sizing with respect to the solution quality.

The remainder of this paper is organized as follows: In Section 2 the related literature is discussed. In Section 3 we present a formal statement of the optimization problem. The Fix-and-Optimize heuristic is described in Section 4. Numerical results are discussed in Section 5. We conclude with some directions for further research in Section 6.

2 Related literature

The option of setup carry-over was first formulated by Dillenberger et al. (1993, 1994), however, without consideration of the trade-off between setup and holding cost which is typical for lot size optimization. Haase (1994) added multi-period setup carry-overs to the CLSP and presented a backward scheduling procedure based on randomized regrets for the resulting CLSP-L. In a later paper, Haase (1998) treated the case of a single-period setup carry-over with a local search heuristic. Gopalakrishnan et al. (1995), Gopalakrishnan (2000) and Gopalakrishnan et al. (2001) developed modeling variants that differ with respect to the modeling of setups. Sox and Gao (1999) proposed a Lagrangean heuristic for a CLSP-L without setup times. Valid inequalities for the CLSP-L are derived by Sürie and Stadtler (2003) and used in a time-oriented decomposition approach with overlapping time windows. A standard MIP solver is applied to the subproblems that are defined during the decomposition. Quadt (2004) and Quadt and Kuhn (2005) use aggregation and decomposition techniques to solve a CLSP-L that is formulated for the bottleneck resource in a multi-stage production system. Gupta and Magnusson (2005) study the case of both sequence-dependent setup times and costs.

The MLCLSP-L extends the MLCLSP through the inclusion of setup carry-overs. The MLCLSP which was introduced by Billington (1983) has been subject to substantial research for more than three decades. Tempelmeier and Helber (1994), Helber (1995) and Tempelmeier and Derstroff (1996) presented early product-oriented decomposition approaches for the MLCLSP with general bill-of-material structures. The heuristic proposed by Tempelmeier and Derstroff (1996) uses Lagrangean re-

laxation to decompose the original problem into a number of uncapacitated single-product Wagner-Whitin problems (see Wagner and Whitin (1958)). This yields a lower bound on the objective function value and a starting point for a heuristic finite scheduling procedure that aims at a feasible production plan and an upper bound on the minimum objective value. A linear programming-based approach that uses coefficient modification is proposed by Katok et al. (1998). A mixed-integer programming-based heuristic with internally rolling schedules is developed by Stadtler (2003). He solves a series of interrelated subproblems with overlapping time windows. The simple plant location formulation of the lot sizing problem is used to speed up the optimization process of the subproblems. The same idea is used for the single-level case treated in Sürie and Stadtler (2003). Stadtler (2003) reports high-quality solutions for small to medium-sized problems. However, as lead times are neglected, it is usually impossible to generate a feasible schedule from an MLCLSP solution. A similar algorithm is presented by Rossi (2003).

Pitakaso et al. (2006) propose to decompose the MLCLSP with respect to products and periods. The form of the decomposition, which results in a sequence of subproblems comprising different sub-rectangles of the product-period matrix, is governed by an ant colony optimization (ACO) meta heuristic. Another population-based "memetic" algorithm is proposed by Berretta and Rodrigues (2004). Tempelmeier and Buschkühl (2008) consider the MLCLSP-L with single-period setup carry-overs and modify the Lagrangean heuristic of Tempelmeier and Derstroff (1996) to deal with this situation.

Helber and Sahling (2008) develop the Fix-and-Optimize approach for the MLCLSP and compare it to the heuristics of Tempelmeier and Derstroff (1996) and Stadtler (2003). The numerical results show that this approach outperforms Stadtler's heuristic with respect to both solution time and quality and the Lagrangean heuristic of Tempelmeier and Derstroff with respect to the solution quality. In the sequel the Fix-and-Optimize approach is adjusted to solve the MLCLSP-L with linked lot sizes and multiple setup carry-overs.

3 Problem statement and model formulation

We consider a general multi-level product structure with several end products. The external demand d_{kt} for product k in period t is given and must be satisfied without backorders. For each product k an initial inventory $\widehat{y}_k$ is given. Each product is produced on a single resource j with limited capacity b_{jt} per period, that can be extended by using overtime O_{jt}. In the heuristic to be presented, overtime is basically used as a slack variable in order to guarantee that the MIP solver always finds a feasible solution. Product k can only be produced in period t if the associated resource has the required product-specific setup state $\gamma_{kt} = 1$. This setup state can either have been carried over from the preceding period $t-1$ or it can result from a new setup operation performed in period t. In the former case, the binary setup carry-over variable ω_{kt} is set to 1. This may happen over multiple consecutive periods. A setup operation may cause setup costs as well as setup time.

Our objective is to determine production quantities Q_{kt} and end-of-period inventory levels Y_{kt} for product k in period t as well as a setup pattern that minimize the sum of setup, holding and overtime costs. In the following mathematical formulation of the optimization model the notation in Table 1 is used.

Table 1: Notation used for model MLCLSP-L

Indices and index sets:	
$\mathcal{J}$	set of resources ($j \in \{1, \ldots, J\}$)
$\mathcal{K}$	set of products ($k \in \{1, \ldots, K\}$)
$\mathcal{T}$	set of periods ($t \in \{1, \ldots, T\}$)
$\mathcal{K}_j$	set of products produced by resource j
$\mathcal{N}_k$	set of immediate successors of product k
Parameters:	
a_{ki}	number of units of product k required to produce one unit of product i (Gozinto factor)
b_{jt}	available capacity of resource j in period t
M_{kt}	big number
d_{kt}	external demand of product k in period t
oc_j	overtime cost per unit of overtime at resource j
h_k	holding cost of product k per unit and period
s_k	setup cost of product k
tp_k	production time per unit of product k
ts_k	setup time of product k
$\widehat{y}_k$	physical initial inventory for product k
Decision variables:	
O_{jt}	overtime at resource j in period t
Q_{kt}	production quantity (lot size) of product k in period t
Y_{kt}	inventory of product k at the end of period t
Y_k^0	initial inventory of product k
γ_{kt}	binary setup state variable of product k in period t
ω_{kt}	binary setup carry-over variable for item k at the beginning of period t
ν_{jt}	auxiliary variable for resource j in period t

Model MLCLSP-L

$$\min Z \quad = \quad \sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}}\left[s_k \cdot (\gamma_{kt} - \omega_{kt}) + h_k \cdot Y_{kt}\right] + \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} oc_j \cdot O_{jt} \quad (1)$$

subject to:

$$Y_{k,t-1} + Q_{kt} - \sum_{i\in\mathcal{N}_k} a_{ki}\cdot Q_{i,t+1} - Y_{kt} = d_{kt}, \qquad \forall k,\, t = 2,\ldots,T-1 \quad (2)$$

$$Y_{k,T-1} + Q_{kT} - Y_{kT} = d_{kT}, \qquad \forall k \quad (3)$$

$$\widehat{y}_k - \sum_{i\in\mathcal{N}_k} a_{ki}\cdot Q_{i1} - Y_k^0 = 0, \qquad \forall k \quad (4)$$

$$Y_k^0 + Q_{k1} - \sum_{i\in\mathcal{N}_k} a_{ki}\cdot Q_{i2} - Y_{k1} = d_{k1}, \qquad \forall k \quad (5)$$

$$\sum_{k\in K_j}\left[tp_k\cdot Q_{kt} + ts_k\cdot(\gamma_{kt}-\omega_{kt})\right] \le b_{jt} + O_{jt}, \qquad \forall j,t \quad (6)$$

$$Q_{kt} \le M_{kt}\cdot\gamma_{kt}, \qquad \forall\, k,t \quad (7)$$

$$\sum_{k\in\mathcal{K}_j}\omega_{kt} \le 1, \qquad \forall j,t \quad (8)$$

$$\omega_{kt} \le \gamma_{k,t-1}, \qquad \forall k,t \quad (9)$$

$$\omega_{kt} \le \gamma_{kt}, \qquad \forall k,t \quad (10)$$

$$\omega_{kt} + \omega_{k,t+1} \le 1 + \nu_{jt}, \qquad \forall j,k\in\mathcal{K}_j,t \quad (11)$$

$$(\gamma_{kt} - \omega_{kt}) + \nu_{jt} \le 1, \qquad \forall j,k\in\mathcal{K}_j,t \quad (12)$$

$$\omega_{k1} = 0, \qquad \forall k \quad (13)$$

$$Q_{kt},\, Y_{kt} \ge 0, \qquad \forall k,t \quad (14)$$

$$Y_k^0 \ge 0, \qquad \forall k \quad (15)$$

$$O_{jt},\, \nu_{jt} \ge 0, \qquad \forall j,t \quad (16)$$

$$\gamma_{kt},\, \omega_{kt} \in \{0,1\}, \qquad \forall k,t \quad (17)$$

The objective function (1) defines the sum of the setup, holding and overtime costs. A setup operation with associated setup costs and setup time for produkt k

occurs in period t if the resource is in the setup state ($\gamma_{kt} = 1$) and there has been no setup carry-over ($\omega_{kt} = 0$).

The inventory balance equations (2) to (5) reflect the planned lead time in the multi-level production structure. From the initial physical inventory $\widehat{y}_k$ for product k only Y_k^0 units enter the balance equations (5) for product k in period $t = 1$ as the remaining units are required by the successors items $i \in \mathcal{N}_k$ in the first period, as described in equation (4). Given the lead time of one period between production stages, it is always possible to transform the setup pattern and the lot sizes into a detailed schedule for the resources. Constraints (6) ensure that the production quantities and setup times meet the capacity constraints for all resources.

The inequalities (7) state that a product can only be produced in a period if its associated resource is in the correct setup state. According to inequalities (8), a setup carry-over into period t is possible for at most one product k at each resource j. Inequalities (9) and (10) require that in case of a setup carry-over, the resource is set up for product k both in periods $t-1$ and t. Constraints (11) and (12) model multi-period setup carry-overs. If the setup state of a resource is carried over from period $(t-1)$ to t and further to period $(t+1)$, then no setup operation must occur in period t. Thus, $\omega_{kt} + \omega_{k,t+1} = 2$ and $\nu_{jt} = 1$. Hence, according to (10), $\gamma_{kt} = 1$, and (12), $(\gamma_{kt} - \omega_{kt}) = 0$, which means that the resource remains in the setup state k.

Equalities (13) state that there is no initial setup state for the resources in the first period. Constraints (14) – (16) are the non-negativity constraints for the production quantity, inventory level, overtime and auxiliary variables. Setup state and carry-over variables are restricted to be binary, according to constraint (17). Formally, this version of the MLCLSP-L is always feasible as there is no limit on overtime. The overtime cost coefficient oc_j can be used to price out undesired overtime from the solution by the algorithm presented in the next section.

The solution of model MLCLSP-L with the help of a MIP solver is usually restricted to very small problem instances, as the linear-programming relaxation of the above "inventory and lot size" formulation provides only relatively weak lower bounds. However, the simple plant location reformulation applied by Stadtler

(2003) to the MLCLSP can also be used to reformulate model MLCLSP-L. We take advantage of this reformulation to compute lower and upper bounds on the objective function values as a benchmark in our numerical study in Section 5.

4 The Fix-and-Optimize Algorithm

4.1 Basic idea

The reasoning behind our algorithm for the MLCLSP-L is that the number $(2|\mathcal{K}|\cdot|\mathcal{T}|)$ of binary setup state and setup carry-over variables γ_{kt} and ω_{kt} determines most of the numerical effort in the Branch&Bound process required to solve the mixed-integer program. By contrast, the number of real-valued variables is of secondary importance. We therefore iteratively solve a series of subproblems $s \in \mathcal{S}$ that are derived from the MLCLSP-L in a systematic manner and that can be solved quickly. In each iteration of the decomposition algorithm we treat one such subproblem $s \in \mathcal{S}$. For each subproblem s, most of the binary setup state variables γ_{kt} are set to a fixed value $\overline{\gamma}_{kt}^{fix}$. The same holds true for the binary setup carry-over variables ω_{kt}, which are set to $\overline{\omega}_{kt}^{fix}$. This leads to a very limited number of non-fixed binary variables which are optimized in a given subproblem. A standard MIP solver is then used to solve the subproblem to optimality. From the solution of the subproblem we determine the new temporary solution for the binary setup variables. The next iteration of the algorithm treats a new subproblem with a different subset of fixed binary variables. However, in each subproblem we consider the complete set of real-valued decision variables. This eliminates the need to freeze a subset of the real-valued decision variables as well as to take care of end-of-horizon effects as in approaches with internally rolling schedules, e. g. the one proposed by Stadtler (2003). Note that our approach differs from a Relax-and-Fix heuristic (Pochet and Wolsey 2006, pp. 109) as we never deal with relaxed binary setup variables in our subproblems.

4.2 Definition of subproblems

In order to define a subproblem, we use the additional notation shown in Table 2.

Table 2: Additional notation for the definition of a subproblem

Sets:	
$(k,t) \in \mathcal{KT}$	set of all product-period combinations
$\mathcal{KT}^{opt}_{\gamma,s} \subseteq \mathcal{KT}$	set of product-period combinations for which binary setup state variables γ_{kt} are optimized in the current subproblem ($\mathcal{KT} = \mathcal{K} \times \mathcal{T}$)
$\mathcal{KT}^{opt}_{\omega,s} \subseteq \mathcal{KT}$	set of product-period combinations for which binary setup carry-over variables ω_{kt} are optimized in the current subproblem
$\mathcal{KT}^{fix}_{\gamma,s} \subseteq \mathcal{KT}$	set of product-period combinations for which binary setup state variables γ_{kt} are fixed in the current subproblem
$\mathcal{KT}^{fix}_{\omega,s} \subseteq \mathcal{KT}$	set of product-period combinations for which binary setup carry-over variables ω_{kt} are fixed in the current subproblem
Parameters:	
$\overline{\gamma}_{kt}$	exogenous value of the fixed setup state variables γ_{kt}
$\overline{\omega}_{kt}$	exogenous value of the fixed setup carry-over variables ω_{kt}
Z_k	estimated costs associated to product k

A subproblem MLCLSP-L-SUB is derived from model MLCLSP-L simply by adding the constraints

$$\gamma_{kt} = \overline{\gamma}_{kt} \qquad \forall\, (k,t) \in \mathcal{KT}^{fix}_{\gamma,s}. \tag{18}$$

$$\omega_{kt} = \overline{\omega}_{kt} \qquad \forall\, (k,t) \in \mathcal{KT}^{fix}_{\omega,s}. \tag{19}$$

This reduces the solution space with respect to the binary variables to the sets $\mathcal{KT}^{opt}_{\gamma,s} = \mathcal{KT} \setminus \mathcal{KT}^{fix}_{\gamma,s}$ and $\mathcal{KT}^{opt}_{\omega,s} = \mathcal{KT} \setminus \mathcal{KT}^{fix}_{\omega,s}$. We are aware that this is not the mathematically most compact form to state the model. However, modern solvers automatically detect and resolve the redundancies of the current formulation.

In our algorithm, we decompose the original problem into ordered sets $s \in \mathcal{S}$ of subproblems. The relative number

$$\Delta_s = \frac{|\mathcal{KT}^{opt}_{\gamma,s}| + |\mathcal{KT}^{opt}_{\omega,s}|}{2 \cdot |\mathcal{KT}|} \tag{20}$$

of non-fixed binary variables in subproblem s determines to which degree the lot sizing decision w. r. t. to products and periods are made simultaneously in an in-

stance of model MLCLSP-L-SUB. The larger Δ_s is for a given instance of problem MLCLSP-L-SUB, the more time-consuming is the solution process and the higher will be the quality of the solution. If an instance s of problem MLCLSP-L-SUB contains only few non-fixed setup variables, i.e, $0 < \Delta_s \ll 1$, these should be selected very carefully, as there should be strong interdependencies between these variables in order to offer trade-offs to be used within the optimization. We propose the following three ways of decomposing the problem into ordered sets $s \in \mathcal{S}$ of subproblems:

- **Product-oriented decomposition:** In this type of decomposition, in a given subproblem s all binary setup state variables γ_{kt} are optimized for a single product k over the complete planning horizon $|\mathcal{T}|$. In addition, we optimize all binary setup carry-over variables $\omega_{\widehat{k}t}$ for all products $\widehat{k} \in \mathcal{K}_{j(k)}$ that require the same resource $j(k)$ as product k.

- **Resource-oriented decomposition:** Here, with respect to the setup state variables γ_{kt}, each subproblem s is defined for a (single) resource j and a time window of four consecutive periods. Unless this time window contains the first or last period, the setup carry-over variables ω_{kt} refer to all the periods within this time window and the one period immediately following the current time window. Assume, for example, that in the current subproblem we are dealing with the setup states in periods 3, 4, 5 and 6. Then we would consider setup carry-overs from periods 3 to 7, again for all products $k \in \mathcal{K}_j$. In two successive subproblems s related to the same resource j we use an overlap of two periods, e.g. periods 1 to 4 for the first subproblem, periods 3 to 6 for the second one etc.

- **Process-oriented decomposition:** With respect to the input-output-relationships between products, in each subproblem s we consider the setup state variables γ_{kt} of product k and one of its immediate successors $i \in \mathcal{N}_k$ for a subset of periods t. For each edge in the BOM graph we define two successive subproblems s such that the first one covers the first half of the planning

horizon and the other one the second half. In addition, we consider the setup carry-over variables ω_{kt} for all products $\widehat{k}$ that require either the same resource as product k or as product k's immediate successor i, i. e. $\widehat{k} \in (\mathcal{K}_{j(k)} \cup \mathcal{K}_{j(i)})$. From these setup carry-over variables we consider the subset defined for the first half of the periods *plus* the immediately succeeding period in the first subproblem (periods 1 to 5 in our example) and the second half (periods 5 to 8) in the second subproblem.

Note that each type of decomposition reflects a specific perspective on the overall optimization problem. The decomposition types are combined in the following four variants of our algorithm:

- **Variant 1:** Product-oriented decomposition only
- **Variant 2:** Product-oriented decomposition first, then resource-oriented decomposition
- **Variant 3:** Product-oriented decomposition first, then process-oriented decomposition
- **Variant 4:** Product-oriented decomposition first, then resource-oriented decomposition, finally process-oriented decomposition

We observed that it can be worthwhile to treat each subproblem more than once, in particular until a local optimum is reached, which suggests an iterative layout of the heuristic.

The product-oriented decomposition is the first step in all four variants of the algorithm. In this type of decomposition, we found it useful to start with those products that are responsible for most of the costs. To compute product-specific costs, we allocate overtime costs proportionally to the capacity requirements of the products. Given that the latter are not known in advance, we estimate product-specific costs Z_k from the solution of the LP-relaxation of problem MLCLSP-L for

each product type as follows:

$$Z_k = \sum_{t\in\mathcal{T}} (s_k \cdot (\gamma_{kt}^{rel} - \omega_{kt}^{rel}) + h_k \cdot Y_{kt}) + \frac{\sum_{t\in\mathcal{T}} \left[tp_k \cdot Q_{kt} + ts_k \cdot (\gamma_{kt}^{rel} - \omega_{kt}^{rel})\right] \cdot \left[\sum_{t\in\mathcal{T}} oc_{j(k),t} \cdot O_{j(k),t}\right]}{\sum_{i\in\mathcal{K}_{j(k)}} \sum_{t\in\mathcal{T}} \left[tp_i \cdot Q_{it} + ts_i \cdot (\gamma_{it}^{rel} - \omega_{it}^{rel})\right]} \tag{21}$$

where γ_{kt}^{rel} and ω_{kt}^{rel} are the optimum values of the binary setup variables γ_{kt} and ω_{kt} in the LP-relaxation and Q_{kt}, Y_{kt} and $O_{j(k),t}$ are the corresponding optimum values of the other decision variables (lot sizes, inventory, overtime). In this cost allocation scheme, the cost of overtime is charged to the products proportionally to the capacity usage of the respective resource. We order the products according to decreasing Z_k in the ordered set $\mathcal{S}$ of subproblems for the product-oriented decomposition.

4.3 The algorithm

The basic structure of our method is outlined in Algorithm 1. At the beginning, we assume that each resource has the required setup state for all products it can possibly produce during each period. Given this setup state pattern, we initially determine production quantities, inventory levels, planned overtime and a first setup carry-over pattern. This initial solution yields an initial objective value Z^{new}. Particularly in the presence of setup times, this solution may be economically unattractive, but we observed that the iterative algorithm prices out unnecessary setups and overtime quickly. After the initialization, the algorithm iterates through the ordered set of subproblems according to the selected variant of the algorithm (1 to 4, see above) either once ($\ell^{max} = 1$) or until it reaches a local optimum ($\ell^{max} = \infty$). Then it terminates and reports the solution that leads to the local optimum Z^{old}.

Note that each solution to a subproblem s yields an objective value Z that is at least as good as the currently best value Z^{old}. For this reason, a new solution is only accepted if it yields lower cost than the currently best solution. We use a boolean variable $CapFeas$ to indicate whether a capacity feasible solution has already been

found. A capacity infeasible solution (with overtime) is never accepted as a new solution if there is a known capacity feasible solution already.

Algorithm 1 Fix-and-optimize algorithm

$\overline{\gamma}_{kt} = 1, \forall (k,t) \in \mathcal{KT}$
$\mathcal{KT}^{fix}_{\gamma,s} \leftarrow \mathcal{KT}$
$\mathcal{KT}^{fix}_{\omega,s} \leftarrow \emptyset$
solve MLCLSP-L-SUB and determine objective function value Z
$Z^{new} = Z$
$\overline{\omega}_{kt} = \omega_{kt}, \forall (k,t) \in \mathcal{KT}$
if $\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} O_{jt} = 0$ **then**
  $CapFeas$= yes
else
  $CapFeas$= no
end if
$\ell = 0$
repeat
  $\ell = \ell + 1$
  $Z^{old} = Z^{new}$
  for each decomposition $\mathcal{S}$ in the current variant of the algorithm **do**
    for each subproblem $s \in \mathcal{S}$ **do**
      determine $\mathcal{KT}^{opt}_{\gamma,s}$ and $\mathcal{KT}^{fix}_{\gamma,s} = \mathcal{KT} \backslash \mathcal{KT}^{opt}_{\gamma,s}$ for subproblem s
      determine $\mathcal{KT}^{opt}_{\omega,s}$ and $\mathcal{KT}^{fix}_{\omega,s} = \mathcal{KT} \backslash \mathcal{KT}^{opt}_{\omega,s}$ for subproblem s
      solve MLCLSP-L-SUB and determine objective function value Z
      if $\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} O_{jt} = 0$ **then**
        $CapFeas^{new}$= yes
      else
        $CapFeas^{new}$= no
      end if
      if $Z < Z^{old}$ and $\{CapFeas^{new}$ or [not($CapFeas^{new}$) and not($CapFeas$)]$\}$
      then
        $\overline{\gamma}_{kt} = \gamma_{kt}, \forall (k,t) \in \mathcal{KT}^{opt}_{\gamma,s}$
        $\overline{\omega}_{kt} = \omega_{kt}, \forall (k,t) \in \mathcal{KT}^{opt}_{\omega,s}$
        $Z^{new} = Z$
        if $CapFeas^{new}$ **then**
          $CapFeas$= yes
        end if
      end if
    end for
  end for
until $\ell = \ell^{max}$ or $Z^{new} \geq = Z^{old}$

5 Numerical Results

5.1 Test sets and reference values

In order to evaluate the solution quality of the proposed algorithm, we used test instances developed and explained in detail by Tempelmeier and Buschkühl (2008). A total number of 1,920 different problem instances with up to 40 products, 16 periods, and 6 resources are grouped into six classes as described in Table 3. A detailed description of the test instances is downloadable from http://www.scmp.uni-koeln.de/publikationen/ORS2008MLCLSPL.zip.

Table 3: Dimensions of the test sets

	Products	Demand periods	Resources	Test instances
Class 1	10	4	3	480
Class 2	10	8	3	480
Class 3	20	8	6	240
Class 4	20	16	6	240
Class 5	40	8	6	240
Class 6	40	16	6	240

We compare our results to the lower and upper bounds obtained by a truncated Branch&Bound method using CPLEX 10.0 on a Unix workstation with eight parallel UltraSPARC-III-processors with 0.9 GHz each. Furthermore, we used the simple plant location reformulation of the MLCLSP-L. In many cases we were unable to compute the optimal solution within a given time limit of one CPU hour for each problem instance. The Fix-and-Optimize algorithm was implemented in Delphi on a 2.13 GHz Intel Pentium Core2 machine, whereby the CPLEX 10.2 callable library was used to solve the different instances of Problem MLCLSP-L-SUB.

Depending on the variant of the algorithm as described in Section 4, in each instance of Class 1, 10% to 60% of the binary setup variables are considered simultaneously, i.e. $0.1 \leq \Delta_s \leq 0.6$ in any one instance of Problem MLCLSP-L-SUB. This fraction of the optimized binary variables decreases for larger problem instances with more products and periods. In Class 2 only 10% to 33.1% of the binary variables were optimized in a single instance of Problem MLCLSP-L-SUB. Furthermore, in Class 3, 10% to 36.3%, and in Class 4, 5% to 9.1% of the binary variables are

determined simultaneously. For Class 5, this range was 2.5% to 21.6% of the binary variables and 2.5% to 10.8% in Class 6.

5.2 Analysis of the numerical results

The numerical results are presented in Tables 4 to 6. We report the following values for the four variants of the algorithm: "ADUB" denotes the average relative deviation of the solution from the upper bound obtained by the truncated Branch&Bound approach as explained above. "ADLB" denotes the average relative deviation of the heuristic solution from the strongest lower bound for the truncated Branch&Bound process. "Feas" is the fraction of problem instances, for which a feasible solution, i. e. a solution without overtime, could be found. Finally, "Time" is the computation time in CPU seconds.

Table 4: Results for Class 1 and Class 2

	Problem Class 1				**Problem Class 2**			
	ADUB [%]	ADLB [%]	Feas [%]	Time [sec]	ADUB [%]	ADLB [%]	Feas [%]	Time [sec]
	Single iteration ($l^{max} = 1$)							
Var 1	1.47	1.47	100.00	0.08	1.38	1.39	100.00	0.18
Var 2	0.18	0.18	100.00	0.11	0.33	0.34	100.00	0.41
Var 3	1.41	1.41	100.00	0.20	1.28	1.28	100.00	0.47
Var 4	0.18	0.18	100.00	0.24	0.33	0.33	100.00	0.70
	Multiple iterations ($l^{max} = \infty$)							
Var 1	0.74	0.74	100.00	0.12	0.83	0.84	100.00	0.29
Var 2	0.12	0.12	100.00	0.18	0.23	0.24	100.00	0.60
Var 3	0.73	0.73	100.00	0.27	0.81	0.81	100.00	0.64
Var 4	0.12	0.12	100.00	0.31	0.22	0.22	100.00	0.88

Table 5: Results for Class 3 and Class 4

	Problem Class 3				**Problem Class 4**			
	ADUB [%]	ADLB [%]	Feas [%]	Time [sec]	ADUB [%]	ADLB [%]	Feas [%]	Time [sec]
	Single iteration ($l^{max} = 1$)							
Var 1	1.34	1.34	100.00	0.51	0.91	1.37	100.00	1.09
Var 2	0.55	0.55	100.00	1.02	0.29	0.74	100.00	3.21
Var 3	1.16	1.16	100.00	1.51	0.78	1.24	100.00	3.11
Var 4	0.52	0.52	100.00	2.10	0.27	0.73	100.00	5.23
	Multiple iterations ($l^{max} = \infty$)							
Var 1	0.81	0.81	100.00	0.83	0.58	1.04	100.00	1.77
Var 2	0.39	0.39	100.00	1.51	0.15	0.61	100.00	4.57
Var 3	0.80	0.80	100.00	1.98	0.56	1.02	100.00	4.18
Var 4	0.37	0.38	100.00	2.67	0.15	0.60	100.00	6.94

Table 6: Results for Class 5 and Class 6

	Problem Class 5				**Problem Class 6**			
	ADUB [%]	ADLB [%]	Feas [%]	Time [sec]	ADUB [%]	ADLB [%]	Feas [%]	Time [sec]
	Single iteration ($l^{max} = 1$)							
Var 1	1.49	1.73	100.00	1.59	0.90	1.55	100.00	3.86
Var 2	0.60	0.83	100.00	2.71	0.32	0.97	100.00	8.57
Var 3	1.00	1.23	100.00	5.20	0.58	1.22	100.00	12.11
Var 4	0.50	0.72	100.00	6.56	0.25	0.89	100.00	16.76
	Multiple iterations ($l^{max} = \infty$)							
Var 1	0.75	0.98	100.00	2.65	0.47	1.12	100.00	5.97
Var 2	0.44	0.67	100.00	4.28	0.24	0.88	100.00	13.03
Var 3	0.65	0.88	100.00	7.00	0.40	1.04	100.00	15.88
Var 4	0.38	0.61	100.00	8.82	0.19	0.83	100.00	23.56

We observe the following: The Fix-and-Optimize Heuristic determined a feasible solution for all test instances. Furthermore, in the case of Variant 1 (product-oriented decomposition) the solution found after a single iteration leads to a maximum average deviation within a problem class of 1.5% from the lowest-known upper bound. Furthermore, a single iteration of Variant 2 (resource-oriented decomposition) results in a better solution quality than multiple iterations of Variant 1. Variant 2 outperforms Variant 3 (process-oriented decomposition). However, a combi-

nation of both (Variant 4) leads to better solutions than Variant 2.

Taking all problem classes together, the average deviation of the solution obtained with Variant 4 from the best-known upper bound is less than 0.5%. As expected, the solution time increases with the number of binary variables. Still, even in Class 6, our solution approach needs only 23.5 CPU seconds on average to generate high-quality solutions.

6 Conclusions and Outlook

We have presented an algorithm to solve the MLCLSP-L with multi-period setup carry-overs. In the underlying model we assume a lead time of one period between production stages in order to guarantee that the production plans can be consistently disaggregated into a detailed schedule. The algorithm was tested against a large set of test problems. It appears that the crucial component of the algorithm is the selection of a good subset of products and periods over which the setup carry-over variables are optimized. If this subset fits well to the respective subset of setup state variables, the Fix-and-Optimize approach appears to work very well.

Future work will address the case of parallel machines, which makes the MLCLSP-L formulation much more relevant for industrial applications. If multiple identical machines within a machine group exist, it may be economically attractive to have some machines continuously setup over several periods for a product with high regular demand while the setup of the other machines producing products with low and erratic demand is frequently changed. It might also be interesting to apply the Fix-and-Optimize approach to other small-bucket lotsizing models and to incorporate the limited capacity of the setup operators in the solution approach. Research in this direction is under way.

References

Bahl, H. C., L. P. Ritzman, and J. N. D. Gupta (1987). Determining lot sizes and resource requirements: A review. *Operations Research 35*(3), 329–345.

Berretta, R. and L. F. Rodrigues (2004). A memetic algorithm for a multi-stage capacitated lot-sizing problem. *International Journal of Production Economics 87*(1), 67–81.

Billington, P. J. (1983). *Multi-Level Lot-Sizing with a Bottleneck Work Center, Ph. D. Dissertation Cornell University*. Cornell.

Brahimi, N., S. Dauzere-Peres, N. M. Najid, and A. Nordli (2006). Single item lot sizing problems. *European Journal of Operational Research 168*(1), 1–16.

Buschkühl, L., F. Sahling, S. Helber, and H. Tempelmeier (2008). Dynamic capacitated lotsizing - a classification and review of the literature on "big bucket" problems.

Dillenberger, C., L. F. Escudero, A. Wollensak, and W. Zhang (1993). On solving a large-scale resource allocation problem in production planning. In G. Fandel, T. Gulledge, and A. Jones (Eds.), *Operations Research in Production Planning and Control*, Berlin, pp. 105–119. Springer.

Dillenberger, C., L. F. Escudero, A. Wollensak, and W. Zhang (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research 75*(2), 275–286.

Drexl, A. and A. Kimms (1997). Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research 99*(2), 221–235.

Gopalakrishnan, M. (2000). A modified framework for modelling set-up carryover in the capacitated lotsizing problem. *International Journal of Production Research 38*(14), 3421–3424.

Gopalakrishnan, M., K. Ding, J.-M. Bourjolly, and S. Mohan (2001). A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management Science 47*(6), 851–863.

Gopalakrishnan, M., D. M. Miller, and C. Schmidt (1995). A framework for modelling setup carryover in the capacitated lot sizing problem. *International Journal of Production Research 33*(7), 1973–1988.

Gupta, D. and T. Magnusson (2005). The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers & Operations Research 32*(4), 727–747.

Gupta, Y. P. and Y. Keung (1990). A review of multi-stage lot-sizing models. *International Journal of Operations & Production Management 10*(9), 57–73.

Haase, K. (1994). *Lotsizing and scheduling for production planning*. Berlin: Springer.

Haase, K. (1996). Capacitated lot-sizing with sequence dependent setup costs. *OR Spektrum 18*(1), 51 – 59.

Haase, K. (1998). Capacitated lot-sizing with linked production quantities of adjacent periods. In A. Drexl and A. Kimms (Eds.), *Beyond Manufacturing Resource Planning (MRP II) – Advanced Models and Methods for Production Planning*, pp. 127–146x. Berlin: Springer.

Helber, S. (1995). Lot sizing in capacitated production planning and control systems. *OR Spektrum 17*, 5–18.

Helber, S. and F. Sahling (2008). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. Technical report, Institut für Produktionswirtschaft, Leibniz Universität Hannover, Königsworther Platz 1, D-30167 Hannover.

Jans, R. and Z. Degraeve (2007). Meta-heuristics for dynamic lot sizing: a review and comparison of solution approaches. *European Journal of Operational Research 177*, 1855–1875.

Karimi, B., S. M. T. F. Ghomi, and J. M. Wilson (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega 31*, 365–378.

Katok, E., H. S. Lewis, and T. P. Harrison (1998). Lot sizing in general assembly systems with setup costs, setup times, and multiple constrained resources. *Management Science 44*(6), 859–877.

Kuik, R., M. Salomon, and L. N. van Wassenhove (1994). Batching decisions: structure and models. *European Journal of Operational Research 75*, 243–263.

Maes, J., J. O. McClain, and L. N. van Wassenhove (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research 53*, 131–148.

Maes, J. and L. van Wassenhove (1988). Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. *Journal of the Operational Research Society 39*(11), 991–1004.

Pitakaso, R., C. Almeder, K. F. Doerner, and R. F. Hartl (2006). Combining exact and population-based methods for the constrained multilevel lot sizing problem. *International Journal of Production Research 44*(22), 4755–4771.

Pochet, Y. and L. A. Wolsey (2006). *Production Planning by Mixed Integer Programming*. Heidelberg et al.: Springer.

Quadt, D. (2004). *Lot-Sizing and Scheduling for Flexible Flow Lines*. Lecture Notes in Economics and Mathematical Systems 546. Springer.

Quadt, D. and H. Kuhn (2005). Conceptual framework for lot-sizing and scheduling of flexible flow lines. *International Journal of Production Research 43*(11), 2291–2308.

Quadt, D. and H. Kuhn (2008). Capacitated lot-sizing with extensions: A review. *4OR: A Quarterly Journal of Operations Research 6*(1), 61–83.

Rossi, H. (2003). *Ein heuristisches Dekompositionsverfahren für mehrstufige Losgrößenprobleme.* Ph. D. thesis, Freie Universität Berlin, Fachbereich Wirtschaftswissenschaft.

Salomon, M., L. G. Kroon, R. Kuik, and L. N. van Wassenhove (1991). Some extensions of the discrete lotsizing and scheduling problem. *Management Science 37*(7), 801–812.

Sox, C. R. and Y. Gao (1999). The capacitated lot sizing problem with setup carryover. *IIE Transactions 31*, 173–181.

Stadtler, H. (2003). Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research 51*(3), 487–502.

Staggemeier, A. T. and A. R. Clark (2001). A survey of lot-sizing and scheduling models. In: *23rd Annual Symposium of the Brazilian Operational Research Society (SOBRAPO).* Campos do Jordao SP, Brazil. S. 938–947.

Sürie, C. and H. Stadtler (2003). The capacitated lot-sizing problem with linked lot sizes. *Management Science 49*(8), 1039–1054.

Tempelmeier, H. (2005). *Material-Logistik. Modelle und Algorithmen für die Produktionsplanung und -steuerung in Advanced Planning-Systemen* (6 ed.). Berlin: Springer.

Tempelmeier, H. and L. Buschkühl (2008). A heuristic for the dynamic multi-level capacitated lotsizing problem with linked lotsizes for general product structures. *OR Spectrum.* DOI 10.1007/s00291-008-0130-y.

Tempelmeier, H. and M. Derstroff (1996). A lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science 42*(5), 738–757.

Tempelmeier, H. and S. Helber (1994). A heuristic for dynamic multi-item multi-level capacitated lotsizing for general product structures. *European Journal of Operational Research 75*, 296–311.

Wagner, M. H. and T. M. Whitin (1958). Dynamic version of the economic lot size model. *Management Science 5*, 89–96.

Wolsey, L. A. (1995). Progress with single-item lot-sizing. *European Journal of Operational Research 86*, 395–401.