International Journal of Production Research
Vol. 00, No. 00, 15 July 2004, 1–27

Combining Population-Based and Exact Methods for Multi-Level Capacitated Lot Sizing Problems

Rapeepan Pitakaso
Ubonrajathanee University, Department of Industrial Engineering, Thailand 34190
Christian Almeder, Karl F. Doerner, Richard F. Hartl
University of Vienna, Department of Management Science, Bruenner Strasse 72,
1210 Vienna, Austria

(Received 00 Month 200x; In final form 00 Month 200x)

We present an ant-based algorithm to solve multi-level capacitated lot sizing problems. We apply a hybrid approach where we use the ant system to optimize the decomposition of the problem into smaller subproblems. These subproblems containing only a few items and periods are solved using CPLEX. Then the overall solution is derived by consolidating the partial solutions. This hybrid approach provides superior results with respect to solution quality in comparison to the existing approaches in the literature.

Keywords: *Ant Colony Optimization, multi-level lot-sizing, material requirements planning, problem decomposition, mixed-integer program.*

1 Introduction

In this work we consider the material requirements planning (MRP) that determines production planning for all items of the same product family or so called multi-level lot-sizing decisions. Multi-level lot-sizing decisions identify when and how much of the products and their components to produce such that setup, production, and holding costs are minimized. Making the right decisions in lot-sizing will affect directly the system performance and its productivity, which is important for a manufacturing firm's ability to remain competitive in the market. Therefore, the development and the improvement of solution procedures for lot-sizing problems is very important.

In our study we focus on the multi-level capacitated lot-sizing problem (MLCLS) which was proved to be NP-hard (Maes and McClain 1991). Therefore we develop a hybrid algorithm where we decompose the problem into several small MLCLS problems. The smaller problems are solved by using the exact

---

This research emerged from the PhD thesis by Rapeepan Pitakaso during his stay at the University of Vienna

formulation and the decomposition is controlled by the use of Ant Colony Optimization (ACO) - a population based metaheuristic. The ACO algorithm is also used to determine which product is considered in which subproblem and so the sequence of products in which the lots are determined. We use the MAX-MIN Ant System (Stützle and Hoos 1997) to find the sequence of products to be planned. Afterwards we decompose the problem and each subset of items is solved using CPLEX.

This approach works very well for medium and large-size problems. In most cases our algorithm can improve the best solutions. Since our algorithm is very complex, there is a little draw back for very small instances. We are able to improve solution quality, but computational times are larger than those of other algorithms.

Maes and McClain (1991) proposed several LP-based heuristics for this problem that are applicable only to serial systems — each item has at most one predecessor and one successor. Tempelmeier and Helber (1994) developed a heuristic procedure that can be applied to general systems, so that each item can have more than one successors and predecessors, and multiple resource constraints are allowed. Tempelmeier and Derstroff (1996) apply a Lagrangean relaxation based heuristic which can be applied to systems with a general and an assembly structure. In an assembly system each item may have several predecessors, but at most one successor. Özdamar and Barbarrosoglu (2000) decompose the global problem (MLCLS) into smaller subproblems and the intensive search capability of the simulated annealing is incorporated into the relaxation design. Stadler (2003) solves the MLCLS problem by reducing the number of periods. He solves the problems on a rolling basis by adding later periods and removing earlier ones.

Berretta and Rodrigues (2004) present a Memetic Algorithm for the problem at hand. Another bio-inspired heuristic has been proposed by Xie and Dong (2002), they proposed a Genetic Algorithm. Kirca and Kökten (1994) developed a heuristic algorithm for a multi-item single level lot-sizing problem which they call item-by-item heuristic. In this algorithm at each iteration step, a set of items that have not yet been scheduled is selected and the production schedule over the planning horizon for this set of items is determined. Berretta *et al.* (2005) describe metaheuristic methods to solve multi-level capacitated lot-sizing problems with non-zero lead times.

Our contribution is threefold:

- We present a successful hybridization of exact and metaheuristic optimization approaches. This is done in way that the metaheuristic controls the problem decomposition process.
- We extended the standard MAX-MIN Ant System with ideas from Evolutionary Algorithms in order to reduce the number of parameters by adapting

them automatically.
- We provide new and efficient algorithm for the MLCLS and obtain many new best solutions for well known test instances.

The paper is organized as follows. In Section 2 the mathematical formulation of the MLCLS is given. In Section 3 the decomposition approach for the different items and periods is presented. The ACO algorithm is described in Section 4. Finally, the experimental framework and the computational results are presented in Section 5. We finish the paper with general conclusions in Section 6.

2 Mathematical formulation

The mathematical model we use in this study is taken from Stadtler (1996). The model is as follows.

Dimensions and indices:

P	number of products in the bill of material
T	planning horizon
M	number of resources
i	item index in the bill of material
t	period index
m	resource index

Parameters:

$\Gamma(i)$	set of immediate successors of item i
$\Gamma^{-1}(i)$	set of immediate predecessors of item i
s_i	setup cost for item i
$c_{i,j}$	quantity of item i required to produce one unit of item j.
h_i	holding cost for item i
$a_{m,i}$	capacity needed on resource m for one unit of item i
$b_{m,i}$	setup time for item i on resource m
$L_{m,t}$	available capacity of resource m in period t
c^o_m	overtime cost of resource m
G	sufficiently large number
$E_{i,t}$	external demand for product i in period t
$I_{i,0}$	initial inventory of item i

Decision variables:

$x_{i,t}$	delivered quantity of item i at the beginning of period t.
$I_{i,t}$	inventory level of item i at the end of period t.
$O_{m,t}$	overtime hours used on resource m in period t
$y_{i,t}$	binary variable indicating whether item i is produced in period t ($y_{i,t} = 1$) or not ($y_{i,t} = 0$)

The problem can then be formulated as a mixed integer program:

$$\min \sum_{i=1}^{P} \sum_{t=1}^{T} (s_i y_{i,t} + h_i I_{i,t}) + \sum_{t=1}^{T} \sum_{m=1}^{M} c_m^o O_{m,t} \tag{1a}$$

subject to the set of constraints

$$I_{i,t} = I_{i,t-1} + x_{i,t} - \sum_{j \in \Gamma(i)} c_{i,j} x_{j,t} - E_{i,t} \qquad \forall i, t \tag{1b}$$

$$\sum_{i=1}^{P} (a_{m,i} x_{i,t} + b_{m,i} y_{i,t}) \leq L_{m,t} + O_{m,t} \qquad \forall m, t \tag{1c}$$

$$x_{i,t} - G y_{i,t} \leq 0 \qquad \forall i, t \tag{1d}$$

$$I_{i,t} \geq 0, O_{m,t} \geq 0, x_{i,t} \geq 0, y_{i,t} \in \{0, 1\} \qquad \forall i, t \tag{1e}$$

The objective function in (1a) minimizes the sum of ordering costs and inventory holding costs for all the items over a predefined planning horizon of length T. Therein also over-time cost are considered when capacity is not enough to produce the dynamic demand. Equation (1b) represents the inventory balance equation. The inventory level of item i in period t equals to the sum of inventory level of i in period $t-1$ plus the production quantity of item i in period t minus the internal demand induced by the parents of item i and the external demand. Capacity constraints (1c) ensure that the resources required to produce the required amount of item i in period t plus the required setup time does not exceed the available resources. Constraint (1d) captures the fact that setup costs are considered whenever a batch is produced, with G being the sum of the remaining demand or any other arbitrarily large number.

For performance reasons it is better to choose G as small as possible. Finally, the usual nonnegativity constraints are denoted in (1e).

3 Decomposition

The whole mixed-integer program (MIP) describing a capacitated multi-level lot-sizing problem can hardly be solved exactly within a reasonable time, if real-world problem sizes are considered. Therefore, our approach is to decompose the problem in several subproblems, which are small enough to be solved exactly. Based on the solution of the subproblems we construct a solution for the original problem. There are two obvious possibilities for decomposition: (i) instead of solving the problem for all periods at the same time, construct subproblems which include only a small number of periods; (ii) split the bill of materials in several parts and solve the problem for each part separately. If we would use either approach (i) or (ii) for a realistic scenario with 50-100 items in the bill of materials and 20-40 number of periods, the subproblems would be still too large to solve them fast enough, especially if the solution of the subproblems is used in an iterative procedure. Therefore we perform a combination of (i) and (ii) to decompose the problem. That means, we select several items from the bill of materials and solve the problem for only a few periods.

Before we decompose, we create a so-called lot-sizing sequence of the products, i.e. we number the items from 1 to P in such a way that all successors of a certain item are in a position before that item. If item i is a successor of item j, then $i < j$. So it is possible to schedule the items one after another starting with item 1. For a general problem there are many different possibilities to sequence the items. In section 4 we will explain how we can find a good sequence using an ant-based algorithm.

To illustrate the decomposition approach, consider a small example of 6 items (already numbered from 1 to 6 according to the above rule) and 9 periods. Each subproblem consists of 3 items and 4 periods. The first subproblem (SP1) contains items 1–3 and periods 1–4. As Figure 1 shows, only part I of the solution of the subproblem is used for composing the final solution of the whole problem. Similar to the principle of planning on a rolling basis, part II is recalculated within subproblem SP2 and parts III and IV are fixed after solving subproblems SP4 and SP5 (products 3–5 and periods 4–7). This overlapping is necessary for consideration of the interdependencies between the items and periods in different subproblems.

[Figure 1 about here.]

It remains to decide, what is a useful number of items I_s and number of

periods T_s for the subproblems, and how large should the overlapping region be. The maximal size of the subproblem is determined by the computational time to solve it. Since the decomposition of the problem is embedded into an iterative solution procedure (see section 4), it is necessary to keep the computational effort for each subproblem small.

Now let us go into more detail for a single subproblem. The mixed-integer formulation of the whole problem (1a)-(1e) cannot be transformed to the subproblems in a straightforward manner. The key aspect is to assign correct capacity limits for each subproblem. If we return to our simple example described in Figure 1, we see, that it is not clear how much of the capacity we should use for subproblem SP1 in order to have enough capacity for subproblem SP4. Since overtime is not limited, there exists always a feasible solution. But if, for example, subproblem SP1 uses all available regular capacity, in subproblems SP4 and SP7 we can produce only in expensive overtime and the solution will be poor. Therefore it is necessary to use a reduced regular capacity for each subproblem in order to reserve some of that capacity for subsequent subproblems. Furthermore it may be the case — since a subproblem represents only a small window of the whole problem — that the demand occurring in a certain subproblem exceeds the available capacity in that time interval, although there would be enough capacity available in other subproblems (in earlier periods). Therefore it is necessary to modify the MIP formulation for the subproblems.

For assigning the right capacity to each subproblem, we have to apply 2 modifications:

(i) We have to reduce the available capacity by the amounts already scheduled for that period in other subproblems.
(ii) We have to modify the capacity needs for production and setup of a certain item, in order to consider the capacity needs of its predecessors.

We will use the following additional notation for describing the adaptations for the subproblems:

k	index of the subproblem.
T_s^k	starting time period of subproblem k.
T_e^k	last time period of subproblem k.
P_s^k	number of first item of subproblem k.
P_e^k	number of last item of subproblem k.
$A_{m,i}^k$	modified capacity needed for production of one unit of item i on resource m.
$B_{m,i,t}^k$	modified capacity needed for setup of production of item i in period t on resource m.
$S_{i,t}^k$	modified setup cost for item i in period t of subproblem k.
$X_{i,t}$	lot size for product i in period t (already determined in previous subproblems).
$Y_{i,t}$	binary variable indicating whether item i is scheduled to be produced in period t (already determined in previous subproblems).
$avC_{m,t}^k$	available regular capacity of resource m in period t for subproblem k

The mixed-integer problem for subproblem k is then:

$$\min \sum_{i=P_s^k}^{P_e^k} \sum_{t=T_s^k}^{T_e^k} \left(S_{i,t}^k y_{i,t} + h_i I_{i,t} \right) + \sum_{t=T_s^k}^{T_e^k} \sum_{m=1}^{M} c_m^o O_{m,t} \tag{2a}$$

subject to (each constraint must hold for all $i = P_s^k, \ldots, P_e^k$, $t = T_s^k, \ldots, T_e^k$, and $m = 1, \ldots, M$)

$$I_{i,t} = I_{i,t-1} + x_{i,t} - \sum_{\substack{j \in \Gamma(i) \\ j < P_s^k}} c_{i,j} X_{j,t} - \sum_{\substack{j \in \Gamma(i) \\ j \geq P_s^k}} c_{i,j} x_{j,t} - E_{i,t} \tag{2b}$$

$$\sum_{i=P_s^k}^{P_e^k} (a_{m,i} x_{i,t} + b_{m,i} y_{i,t}) \leq L_{m,t} + O_{m,t} - \sum_{i=1}^{P_s^k - 1} (a_{m,i} X_{i,t} + b_{m,i} Y_{i,t}) \tag{2c}$$

$$\sum_{\tau=T_s^k}^{t} \sum_{i=P_s^k}^{P_e^k} \left(A_{m,i}^k x_{i,\tau} + B_{m,i,\tau}^k y_{i,\tau} \right) \leq \sum_{\tau=T_s^k}^{t} \left(avC_{m,\tau}^k + O_{m,\tau} \right) \tag{2d}$$

$$x_{i,t} - Gy_{i,t} \leq 0 \tag{2e}$$

$$I_{i,t} \geq 0, \quad x_{i,t} \geq 0, \quad O_{m,t} \geq 0, \quad y_{i,t} \in \{0,1\} \tag{2f}$$

It has been known for a long time that simple item by item decomposition gives poor lot-sizing results if the setup costs are not modified (Blackburn and Millen 1982). There are several modification schemes that add part of the setup costs of predecessors to the setup costs of each item. We follow the approach by Pitakaso *et al.* (2005) which is an extension of Dellaert and Jeunet (2003). To the setup cost of an item we add a fraction of the setup costs of all its predecessors, depending if a new setup is necessary for that predecessor or not. For that purpose we use a binary variable $T(i,j,t)$ which takes the value of 1 if a lot-size of item i, delivered in period t, generates a new lot for item j in period t, and it equals zero if there is already a positive demand for item j in period t. This demand results from a planned lot-size for a different successor of item j or is an external demand. In order to avoid adding the setup cost of a single item several times, the modified setup costs are divided by the number of immediate successors in the current subproblem. Hence, in the objective (2a) we do not use the original setup cost, but modified ones as follows:

$$S^k_{i,t} = s_i + r_i \sum_{\substack{j \in \Gamma^{-1}(i) \\ j > P^k_e}} T(i,j,t) \frac{S_j}{|\Gamma(j)|} \tag{3a}$$

where S_j can be calculated recursively by

$$S_i = s_i + \sum_{j \in \Gamma^{-1}(i)} \frac{S_j}{|\Gamma(j)|}, \tag{3b}$$

and

$$r_i = R \cdot \left(1 + \frac{P - 2\Phi_i + 1}{P - 1} \cdot u\right) \tag{3c}$$

with two parameters $R \in U[0, 0.5]$ and $u \in U[-1, 1]$ randomly chosen according two a uniform distribution, and Φ_i denoting the position of item i in

the lot-sizing sequence. For more details on this type of cost modification see Pitakaso *et al.* (2005).

The inventory balance equation (2b) is identical to (1b). Note that for items i with $i < P_s^k$ the lots were already fixed in some previous subproblems, whereas items j with $j \geq P_e^k$ will be considered in later subproblems.

The capacity constraint (2c) ensures that we have to use overtime if the production of a certain period exceeds the free capacity. The total capacity is reduced by the amounts of resources needed to produce the items already scheduled in previous subproblems. This reduction is represented by the last summation term, where we sum up over all already scheduled items.

The cumulative capacity constraint (2d) should ensure that our global solution consisting of the solution of the subproblems will not use overtime if it is not necessary. The left-hand side represents the accumulated capacity needs for all products of the subproblem from the starting period T_s^k of the subproblem up to period t ($t = T_s^k, \ldots, T_e^k$). The modified capacity needs $A_{m,i}^k$ and $B_{m,i,t}^k$ consider that if we schedule a certain item, we need also resource to make the predecessors in some period before. These modified capacity needs can be calculated recursively starting with the last item in the lot-sizing sequence.

$$A_{m,i}^k = a_{m,i} + \sum_{\substack{j \in \Gamma^{-1}(i) \\ j > P_e^k}} A_{m,j}^k, \tag{4a}$$

$$B_{m,i,t}^k = b_{m,i} + \sum_{\substack{j \in \Gamma^{-1}(i) \\ j > P_e^k}} T(i,j,t) \frac{\tilde{B}_{m,j}}{|\Gamma(j) \cap \Delta(i)|}, \tag{4b}$$

where $\tilde{B}_{m,j}$ is the accumulated capacity need for setup for product i on resource m

$$\tilde{B}_{m,i} = b_{m,i} + \sum_{j \in \Gamma^{-1}(i)} \tilde{B}_{m,j}. \tag{4c}$$

The idea of equation (4b) is similar to the setup cost modification in (3a). The expression $|\Gamma(j) \cap \Delta(i)|$ in (4b) ensures that the setup resource needs are divided by the number of immediate successors which are in the same group as item i. These groups are: (i) already lot-sized items; (ii) items in the current

subproblem k; (iii) not yet scheduled items in other subproblems.

$$\Delta(i) = \begin{cases} \{1, \ldots, P_s^k - 1\} & \text{if } i < P_s^k \\ \{P_s^k, \ldots, P_e^k\} & \text{if } P_s^k \leq i \leq P_e^k \\ \{P_e^k + 1, \ldots, P\} & \text{if } i > P_e^k \end{cases} \tag{4d}$$

The available capacity $avC_{m,t}^k$ is based on the total capacity diminished by the resources already used for other products and resources which are reserved for items not scheduled yet.

$$\begin{aligned} avC_{m,t}^k = L_{m,t} &- \sum_{i=1}^{P_s^k - 1} \left(A_{m,i}^k X_{i,t} + B_{m,i,t}^k Y_{i,t} \right) \\ &- \sum_{\substack{i = P_e^k + 1 \\ \Gamma(i) = \emptyset}}^{P} \left(A_{m,j}^k E_{i,t} + B_{m,i,t}^k Y_{i,t}^E \right) \end{aligned} \tag{5}$$

where

$$Y_{i,t}^E = \begin{cases} 1 & E_{i,t} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5a}$$

The first sum in equation (5) represents the amount of resources needed for the already scheduled items 1 to $P_s^k - 1$ including the resource that will be needed for its predecessors which are not scheduled yet and which are not included in the current subproblem. The second sum gives the resources needed for end-items (and their predecessors) which are not scheduled yet and which are not in the current subproblem. Now $avC_{m,t}^k$ considers all already reserved resources and capacity needs caused by already scheduled items. But it is not necessary and not possible to schedule the production for all predecessors in that period t, although they are included. Therefore we use only a cumulative capacity constraint (2d), which allows to shift the production between periods. So we ensure that enough capacity is reserved in any of the previous periods up to the current one.

With the above adaptations for the capacity needs and the available regular capacity we reserve capacity for items which are scheduled later. A similar adaption is necessary to ensure that the regular capacity is used in periods with low demand in order to cover periods with high demand. Since the subproblems include only some but not all periods, it may be necessary to balance such demands in different subproblems. For that purpose we develop a method

for production backward shifting (cf. Berretta and Rodrigues 2004, Franca *et al.* 1994, Trigeiro *et al.* 1989, Xie and Dong 2002). We consider only demand shifting between subproblems of the same level, i.e. subproblems which contains the same items, but different periods.

We always start with the last subproblem of the current level which contains the last period. We take constraint (2d) and substitute the external demands $E_{i,t}$ and the internal demands $\sum_{j\in\Gamma(i)} c_{i,j}X_{j,t}$ for the production quantities and set setups $Y_{i,t}$ if necessary. We check the constraint for each t starting with the first period in the subproblem. If there is no period where we have to use overtime, then it is not necessary to shift any demands. Otherwise we shift the demand of as many units as necessary of the last item in the subproblem (i.e. with the highest number) to the nearest period before, that is outside the current subproblem. If there is not enough demand that can be shifted in order to prevent using overtime, we shift also demands for the next item in the list. After that we analyze the previous subproblem on that level until we reach the subproblem containing period 1, where we stop. This procedure has to be repeated for each resource m.

So the overall procedure for solving the problem by decomposition is as follows:

- Step 1: For each subproblem decide which items and which periods are included. Set the current level of subproblems p to 1.
- Step 2: Perform the capacity modifications (3a)-(5a) for all subproblems of the level p.
- Step 3: Apply the demand shifting procedure for level p and adapt the demands for each subproblem.
- Step 4: Solve each subproblem of the current level p, starting with the first one containing period 1 (e.g. SP1 for level 1). Fix the solution in the non-overlapping region and use the solution (inventory levels) to solve the next subproblem, and so on.
- Step 5: Fix the solution for the non-overlapping items of level p and calculate the new demands for the next level. Stop if p is the last level or set $p := p+1$ and proceed with step 2.

4 ACO Algorithm

In Section 3 we have developed an algorithm for lot-sizing the products based on a given sequence of products. In the next step we propose a MAX-MIN Ant System (MMAS) algorithm developed by Stützle and Hoos (1997, 2000) for optimizing this sequence in order to minimize the total costs.

The main idea of Ant Colony Optimization (ACO) is that a population of artificial ants repeatedly builds and improves solutions to a given instance

of a combinatorial optimization problem (see Colorni *et al.* 1992, Dorigo and Stützle 2004). From one generation to the next a global memory is updated that guides the construction of solutions of the successive population. The memory update is based on the solution quality of the ants and is biased towards the best solutions found. In the MAX-MIN Ant System only the overall best solution found so far is allowed to update pheromone. After the construction phase of the algorithm a local search may be applied to the solutions of the ants. A convergence proof for the ACO algorithm can be found in Gutjahr (2003), Stützle and Dorigo (2002).

Within the ACO algorithm an artificial memory - so called pheromone information is used. After the initialization of this joint memory the standard ACO algorithm mainly consists of the iteration of three steps:

- Step 1: Construction of solutions by ants according to heuristic and pheromone information
- Step 2: Application of a local search to the ants' solutions
- Step 3: Update of the pheromone information

This metaheuristic algorithm has also been used for the uncapacitated problem. For this problem class it outperformed all the existing approaches (see Pitakaso *et al.* 2005). There we applied already ideas from Evolutionary Algorithms to "learn" which R and u in the cost adjustment formula (3a) should be chosen. We now extend this idea to determine how many items I_s and how many periods T_s should be put together in one subproblem.

Each ant generates a production sequence. As heuristic information η_j (also called visibility in the ant system framework) the original setup costs s_j are used (see (6)). We tested different alternative values as heuristic information (combinations of setup costs and holding costs) and also the use of no heuristic information. It turned out that the best results can be reached by using the original setup costs s_j. We do not apply a local search procedure because the calculation of the total cost through the use of the decomposition algorithm is very time consuming. Moreover, the results in Section 5 indicate that it is not necessary to include a local search procedure. So Step 2 does not exist in our algorithm. The pseudo code in Figure 2 illustrates the adaptations of the MMAS to solve the MLCLS. The Ant System for multi-level capacitated lot-sizing problems is denoted as ASMLCLS in the sequel.

[Figure 2 about here.]

The algorithm starts with the initialization phase. In this phase the standard initialization routines like initializing the pheromone matrix are performed. Within this phase the values for R_b, u_b, T_s^b, I_s^b which provide the best solution quality out of 20 randomly generated solutions are selected as initial values.

After the initialization phase the construction phase is executed. In this phase, each ant generates a sequence of items which is used afterwards to apply the decomposition algorithm (see Section 3). After all the ants have constructed their solution and the solution qualities are computed, the pheromone update is applied. In the MMAS the best solution found so far by the ants is allowed to update the artificial pheromone.

We applied the pheromone encoding scheme which provides the best results in the uncapacitated version (see Pitakaso *et al.* 2005). This pheromone encoding scheme was developed for scheduling problems by Stützle (1998) and was also used by Merkle *et al.* (2002). In this pheromone encoding τ_{pj} denotes the desirability of lot-sizing item j as the p-th item. This value gives information how desirable it was to place item i on position p in the previous iterations. This information is used in the construction step (Step 1) of the algorithm. The pheromone information is modified in Step 3 of the algorithm.

So we consider a MLCLS with P products and initial setup costs s_j. Each ant constructs a production sequence (see Step 1). At each step in the construction phase an ant decides independently which item will be lot-sized at the current position of the sequence. The probability that an item is selected on a particular position is determined according to the random proportional rule (6).

$p_{pj}^{\kappa}(\ell)$	probability that ant κ selects product j on position p in iteration ℓ
$\tau_{pj}(\ell)$	intensity of pheromone trail of product j in position p at iteration ℓ
α	parameter to regulate the influence of $\tau_{pj}(\ell)$
β	parameter to regulate the influence of s_j
N_p^{κ}	set of selectable products in position p of ant κ based on the bill of materials

$$p_{pj}^{\kappa}(\ell) = \begin{cases} \frac{[\sum_{o=1}^{p} \tau_{oj}(\ell)]^{\alpha}[s_j]^{\beta}}{\sum_{l \in N_p^{\kappa}}[\sum_{o=1}^{p} \tau_{ol}(\ell)]^{\alpha}[s_l]^{\beta}} & \text{if } j \in N_p^{\kappa} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

For the pheromone used in the decision rule we do not just consider the current pheromone value of setting item j in the current position p. Rather, we consider also all the pheromone values of setting item j in all the predecessor positions of p. We denote the decision rule where this pheromone usage scheme is used as summation decision rule in the following. This summation rule was introduced by Merkle and Middendorf (1999).

As usual in MMAS, the best ant updates the pheromone, but the values are bounded to the interval $[\tau_{min}, \tau_{max}]$:

$\rho \in [0,1]$	trail persistence parameter to regulate the evaporation of τ_{pj}
$\Delta\tau_{pj}(\ell)$	total increase of trail level on edge (p,j) which is controlled by maximum and minimum value along with the concept of MMAS
$f(s^{opt})$	global best solution value

$$\tau_{pj}(\ell+1) = \max\Big(\tau_{min}, \min\big(\tau_{max}, \rho\tau_{pj}(\ell) + \Delta\tau_{pj}(\ell)\big)\Big). \tag{7}$$

with

$$\Delta\tau_{pj}(\ell) = \begin{cases} \frac{1}{f(s^{opt})} & \text{if item } j \text{ is on position } p \text{ for the best ant} \\ 0 & \text{otherwise.} \end{cases}$$

The pheromone bounds are taken from Stützle and Hoos (1997).

An open question is how to select R, u, I_s, T_s for lot-sizing the products and determining the costs of the sequence the ant found. We could choose these parameters randomly. But tests have shown that a systematic search of optimal values for R, u, I_s, T_s gives better solutions (for the results concerning R and u see Pitakaso *et al.* 2005).

This systematic search for good values for R, u, T_s, and I_s is done as follows. We augmented the standard MMAS with the component of selecting R, u, T_s, and I_s by using ideas of evolutionary strategies. For the first iteration we generate randomly a set of different quadruples (R, u, T_s, I_s) and select the best one according to the objective function. For all other iterations we take the basic parameter set (R_b, u_b, I_s^b, T_s^b) representing the best values of the previous iteration. Each ant chooses randomly for its (R, u, I_s, T_s) either this basic set or a slightly perturbed one (see Figure 2). The best solution of each iteration determines the new (R_b, u_b, I_s^b, T_s^b). In some pre-tests we have shown that this procedure gives better solutions than choosing R_b, u_b, I_s^b, and T_s^b constant by taking the best values from the initial phase. In a previous work (see Pitakaso *et al.* 2005) it turned out that for the perturbation rate ϑ a value of $\vartheta = 0.05$ is reasonable (see Figure 2).

5 Results

We implemented the ASMLCLS algorithm in C (Visual C 6) using CPLEX 9.0 for solving the subproblems. The tests were performed on a personal computer *Pentium 4 2.4 GHz* with *1 GB RAM* and *Microsoft Windows 2000*. On the

bases of some test instance we determined limits for the subproblem size such that we were able to solve it on average within 2 seconds. Table 1 shows these limits. If we were not able to find the optimal solution for a specific subproblem within that time, we took the best solution CPLEX found so far. Furthermore, we generated solutions for several test instances with different overlapping factors. It turned out that the best combination is to use 20% overlapping for the items and 60% overlapping for the periods.

[Table 1 about here.]

In total we tested our algorithm with 4 different groups of test instances. The first group of test instances was taken from Berretta and Rodrigues (2004). This group consists of 160 test instances with two different product structures (general (G) and serial (S) systems), with two different levels of setup costs (low (L) and high (H) costs), and four different combinations of number of items (I) and planning horizons (T). Table 2 shows the results of our algorithm compared with the method introduced in Berretta and Rodrigues (2004) and with CPLEX solving the whole problem at once with a time limitation of one hour. For nearly all instances we were able to calculate the optimum, and for the others the gap is extremely small. Our approach needs more computational time than the method of Berretta and Rodrigues (2004)[1] but on average we are more than 50 times faster than CPLEX.

[Table 2 about here.]

The other 3 groups of test instances are from Tempelmeier and Derstroff (1996). Group 2 consists of 600 test instances with 10 items, 4 periods, and 3 resources. There are instances with general systems (G) and instances with assembly systems (A) and two different ways of assigning the resources. In the non-cyclic case (NC), for each production level just 1 resource is needed, whereas in the cyclic case (C) several resources are needed within the same production level. Furthermore the test instances differ in the demand patterns. Table 3 shows a similar picture than before. With ASMLCLS it is possible to find nearly all optimal solutions and the gap for the remaining ones is very small. In Özdamar and Barbarosoglu (2000) another approach integrating Lagrangian relaxation and Simulated Annealing was applied to these test instances. They report an overall mean deviation of 4.21% with an average calculation time of 8.54 seconds. Tempelmeier and Derstroff did their test in 1996 on a computer which is about 1000 times slower than the computer we

[1]The results of Berretta and Rodrigues (2004) were evaluated on a Sun UltraSPARC 60. Since the computational speed of this machine is not included in the paper of Dongarra (2005), we used the similar Sun UltraSPARC 80 for comparison. Hence, the computer we used is approximately 5.7 times faster.

are using (according to Dongarra 2005). This means their method is extremly fast, but they reported in the paper that it was not possible to improve the results significantly if they would have done more iterations.

[Table 3 about here.]

Group 3 of the test instances consists of 600 different problems with 40 items, 16 periods, and 6 resources. Again, there is a cyclic and a non-cyclic resource assignment and different demand patterns. Table 4 shows that ASMLCLS is able to beat the method of Tempelmeier and Derstroff (1996) in 86% of the instances.

[Table 4 about here.]

Group 4 includes 150 instances with 100 items spread over 10 levels, 16 periods, and 10 resources all with a general product structure. There are 5 different levels of the capacity utilization (1-5) and again cyclic and non-cyclic instances. We compare our results with those of Tempelmeier and Derstroff (1996) and Stadtler (2003). For these large test instances the method of Tempelmeier and Derstroff is extremely fast, but has very low solution quality. The method of Stadtler is very complex consuming a lot more computational time, but the results from Tempelmeier and Derstroff (1996) were improved by more than 5%. In Table 5 we present two different results for our algorithm. First, we use 20 minutes run time which correspond to the computation time of Stadtler (2003). We adapted the computational times according to the factors given in Dongarra (2005). Second we allowed to run the algorithm for 30 minutes, and in more than 90% of the test instances ASMLCLS could reach or improve the best solutions.

[Table 5 about here.]

As a last test we investigated if the difference between ASMLCLS and the approach of Tempelmeier and Derstroff (1996) is statistically significant. Furthermore we checked if each of our solution steps improves the solution quality. For that purpose we selected 50 random test instances from group 3 and compared the solution quality given in Tempelmeier and Derstroff (1996) with the following solution methods:

Decomp: Apply the decomposition method with a random lot-sizing sequence (considering the preference rule), and a randomly chosen subproblem size and randomly chosen R and u. We do not apply the demand shifting procedure.
Decomp-Ants: The same as before, but we use the ant algorithm to get a lot-sizing sequence.
Decomp-Ants-Evo: Now the ants also search for the best subproblem

size and the best R and u.

ASMLCLS: The full algorithm including the demand shifting procedure.

We solved the 50 test instances with all variants using the same computational time. We applied the Wilcoxon signed rank test for paired data with 1% threshold. Each step of our algorithm improves the solution quality significantly and all variations are significantly better than the algorithm of Tempelmeier and Derstroff (1996). Furthermore, the complete algorithm (ASMLCLS) is also significantly better than the algorithm by Stadtler (2003).

6 Conclusions

A hybrid approach combining a MMAS, Evolutionary Strategies, and exact solvers for mixed-integer problems has been proposed to solve multi-level capacitated lot-sizing problems. This mixture allows to improve the solution quality for standard test instances. Because of the complex structure of the algorithm, the computational overhead for small problems increases the time necessary to gain a good solution. Nevertheless, it is possible to reach a high solution quality. For large problems the effects of the overhead are reduced, and ASMLCLS is able to outperform the best algorithms. Due to its construction, the algorithm shows its best performance for problems with the following properties:

- General or assembly systems allow more freedom for the ants part to select between different sequences.
- If no overtime is necessary, the shifting demand procedure is faster and the decomposition methods gives better results.
- For large problems the computational overhead of the algorithm has a lower impact on the overall calculation time.

Acknowledgments

We would like to thank Regina Berretta, Horst Tempelmeier, and Hartmut Stadtler for providing the test data to us. Furthermore, we are grateful to Martin Romauch for helpful comments concerning the usage of CPLEX. Finally, financial support by the Austrian Exchange Office (ÖAD) to first author is gratefully acknowledged.

REFERENCES

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

Berretta, R., Franca, P.M., and Armentano, V.A., "Metaheuristic approaches for the multilevel resource-constraint lot-sizing problem with setup and lead times". *Asia-Pacific Journal of Operational Research* **22** (2005) 261–286.

Blackburn, J.D., and Millen, R.A., "Improved heuristic performance in multistage lot sizing systems". *Management Science* **28** (1982) 44–56.

Colorni, A., Dorigo, M. and Maniezzo, V., (1992) "Distributed Optimization by ant colonies", in: *Toward a Practice of Autonomous Systems: Proceedings of The First European Conference on Artificial Life*, The MIT Press.

Dellaert, N.P., and Jeunet, J., "Randomized multi-level lot sizing heuristics for general product structures". *European Journal of Operational Research* **148** (2003) 211–228.

Dongarra, J.J., "Performance of various computers using standard linear equations software, (Linpack Benchmark Report)". *University of Tennessee Computer Science Technical Report*, **CS-89-85** (2005).

Dorigo, M., and Stützle, T., *Ant Colony Optimization*. MIT Press, Cambridge, USA (2004).

Franca, P.M., Armentano, V.A., Berretta, R.E., and Clark, A.R., "A heuristic method for lot sizing in multi-stage systems". *Computers and Operations Research* **24**(9) (1994) 861–874.

Gutjahr, W.J., "A generalized convergence result for the graph-based ant system metaheuristic". *Probability in the Engineering and Informational Sciences* **17** (2003) 545–569.

Kirca, O., and Kökten, M., "A new heuristic approach for the multi-item dynamic lot sizing problem". *European Journal of Operational Research* **75** 1994 332-341.

Maes, J., and McClain, J.O., "Multilevel capacitated lot sizing complexity and LP-based heuristics". *European Journal of Operational Research* **53** (2) (1991), 131–148.

Merkle, D., and Middendorf, M., "An ant algorithm with a new pheromone evaluation rule for total tardiness problems". In: *Boers, E. J.-W. et al., Applications of Evolutionary Computing: EvoWorkshops 2001*. Springer LNCS 2037, Berlin, (1999) 287–296.

Merkle, D., Middendorf, M., and Schmeck, H., "Ant Colony Optimization for resource constrained project scheduling", *IEEE Transactions on Evolutionary Computation* **6** (2002) 333–346.

Özdamar, L., and Barbarosoglu, G., "An integrated Lagrangean relaxation-simulated annealing approach to the multi-level multi-item capacitated lot sizing problem." *International Journal of Production Economics* **68** (2000) 319–331.

Pitakaso, R., Almeder, C., Doerner, K., and Hartl R. F., "A MAX-MIN ant system for the uncapacitated multi-level lot sizing problem". to appear: *Computers & Operations Research* (2005).

Stadtler, H., "Mixed integer programming model formulations for dynamic multi-item multi-level capacitated lotsizing". *European Journal of Operational Research* **94** (1996) 561–581.

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

Stützle, T., "An ant approach to the flow shop problem". *Proceedings of EUFIT'98*, Aachen (1998) 1560-1564.

Stützle, T., and Hoos, H., "The MAX-MIN ant system and local search for the traveling salesman problem". *Proceedings of ICEC'97 – 1997 IEEE 4th International Conference on Evolutionary Computation*, IEEE Press (1997) 308–313.

Stützle, T., and Hoos, H., "MAX-MIN ant system". *Future Generation Computer Systems* **16** (2000) 889–914.

Stützle, T., and Dorigo, M., "A short convergence proof for a class of ACO algorithms." *IEEE Transactions on Evolutionary Computation* **6** (4) (2002) 358-365.

Tempelmeier, H., and Derstroff, M., "Mehrstufige Mehrprodukt-Losgrößen-Planung bei beschränkten Ressourcen und genereller Erzeugnisstruktur". *OR Spektrum* **15** (1993) 63–73.

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

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

Trigeiro, W., Thomas, L.J., and McClain, J.O., "Capacitated lot sizing with set-up times". *Management Science* **35**(3) (1989) 738–757.

Xie, J. and Dong, J., "Heuristic genetic algorithms for general capacitated lot-sizing problems" *Computer and Mathematics with Applications* **44** 2002 263–276.

Figures

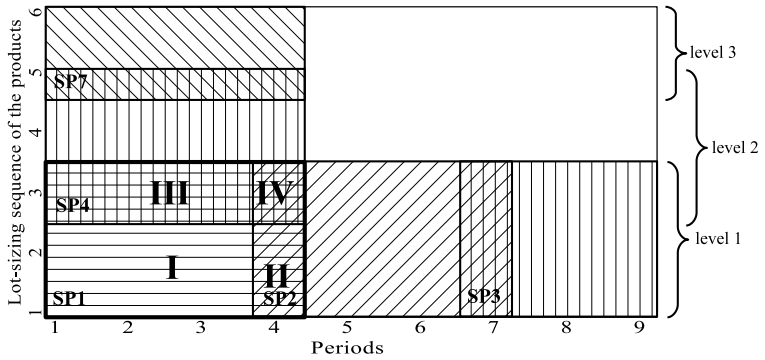

Figure 1. Scheme of subproblems with overlapping of one period and one product. After solving subproblem 1 (SP1), only part I is taken for the final solution, the other parts are recalculated within other subproblems due to the overlapping. SP1-SP3 form the first decomposition level

Procedure ASMLCLS

```
/* Initialization phase */
Generate initial R_b,u_b, T_s^b, and I_s^b
(select best solution out of 20 randomly constructed ones)
Initialize pheromone information
while (termination condition not met) do
 for each ant do
   /* Construction phase (Step 1)*/
   Construct the production sequence according to the decision rule (6).
   /* Adaptation of R and u values for each ant */
   Choose R randomly out of the set {R_b(1 − ϑ), R_b, R_b(1 + ϑ)}
   Choose u randomly from {max{−1, u_b(1 − ϑ)}, u_b, min{1, u_b(1 + ϑ)}}
   Calculate r_i according to equation (3c)
   Choose I_s randomly out of the set {I_s^b − 1, I_s^b, I_s^b + 1}
   Choose T_s randomly out of the set {T_s^b − 1, T_s^b, T_s^b + 1}
   (within the boundaries of table 1)
   Perform the decomposition method of Section 3 to evaluate the sequence
 end
 /* Pheromone update phase (Step 3) */
 Update the pheromone matrix according to (7), update R_b, u_b, T_s^b, I_s^b
end
```

Figure 2. Pseudo code of ASMLCLS

Tables

Table 1. The maximal subproblem size which allows to get a solution within 2 seconds. I_s indicates the number of items included in the subproblem and T_s is number of periods for that subproblem.

I_s	1	2	3	4	5	6	7	8	9	10	11	12
T_s	15	13	10	9	8	6	5	5	4	3	3	2

Table 2. Comparison of the results for the test instances from Berretta and Rodrigues (2004). Column (B&R) contains the results of Berretta and Rodrigues (2004); column CPLEX refers to CPLEX results within 1 hour computation time; column ASMLCLS contains our results. For problems marked with * there is no optimal solution available, but the last column shows the result of CPLEX solved within a tolerance of 1%. MAPD indicates the mean absolute percentage deviation of the method in per cent to this optimal (or nearly optimal) solution.

Problem	B&R		CPLEX (3600s)		ASMLCLS			(Optimal)
	MAPD	sec.	MAPD	sec.	MAPD	sec.	Cost	Cost
G-L-3I-6T	0	1.45	0	8.71	0	10.4	4502.02	4502.02
G-L-3I-12T*	0.1	4.98	0.032	2753.09	0	13.5	9383.62	9383.62
G-L-6I-6T	0	4.02	0	0.03	0	21.1	12140.11	12140.11
G-L-10I-6T*	-0.1	18.16	0	46.26	0.0027	24.2	18461.73	18461.26
G-H-3I-6T	0.1	1.27	0	763.61	0	10.8	9149.52	9149.52
G-H-3I-12T*	0.3	3.79	0.76	3600.00	0.0057	13.5	16726.87	16725.71
G-H-6I-6T	0.3	2.64	0	0.36	0	25.5	23148.57	23148.57
G-H-10I-6T*	0.4	8.06	0	1702.08	0.0027	24.3	31750.07	31749.18
S-L-3I-6T	0.1	0.84	0	6.85	0	10.4	3390.35	3390.35
S-L-3I-12T*	0.2	2.77	0.18	1826.49	0	13.4	7218.09	7218.09
S-L-6I-6T	0.1	2.07	0	6.27	0	23.0	7524.13	7534.13
S-L-10I-6T*	0.1	6.02	0	1600.39	0	23.7	12793.65	12793.65
S-H-3I-6T	0.6	0.85	0	3.85	0	11.0	7525.16	7525.16
S-H-3I-12T*	0.4	2.27	0.19	2668.18	0.0062	13.5	13569.57	13568.84
S-H-6I-6T	0.4	1.63	0	7.61	0	25.4	17424.97	17424.97
S-H-10I-6T*	0.1	3.76	0	1476.69	0	24.6	25953.86	25953.86
Mean	0.2	4.04	0.0726	1029.41	0.0011	18.0	13792.02	13791.81

Table 3. For group 2 of the test instances from Tempelmeier and Derstroff (1996) we compare in this table our results (ASMLCLS) with those of Tempelmeier and Derstroff (1996) (T&D) and Berretta and Rodrigues (2004) (B&R). MAPD indicates the mean deviation of the method in per cent to the optimal solution. % is the percentage of optimal solutions found.

Problem	T&D			B&R			ASMLCLS				Optimal
	MAPD	%	sec.	MAPD	%	sec.	MAPD	%	sec.	Cost	Cost
G-NC	2.0	46		0.2	91	30.3	0.005	93.3	28.03	11163.9	11163.5
G-C	1.8	41		0.4	83	30.2	0.016	89.3	28.01	11334.6	11332.8
A-NC	0.7	62		0.4	83	30.3	0.006	92.0	27.50	10803.0	10802.5
A-C	0.7	62		0.1	96	29.6	0.002	96.7	27.25	10347.2	10347.1
Mean	1.3	53	1.45	0.3	89	30.1	0.007	92.8	27.54	10912.2	10911.5

Table 4. For group 3 of the test instances from Tempelmeier and Derstroff (1996) we compare in this table our results (ASMLCLS) with those of Tempelmeier and Derstroff (1996) (T&D). MAPD indicates the mean deviation of the method in per cent to the best solution. % is the percentage of best solutions found. The last column shows the best solutions available.

Problem	T&D				ASMLCLS				Minimal
	MAPD	%	min.	Cost	MAPD	%	min.	Cost	cost
G-NC	0.924	30.7		382706	0.049	84.7	10.8	380666	380512
G-C	0.865	29.3		395075	0.053	88.7	10.6	393329	393208
A-NC	2.324	25.3		47134	0.092	82.6	9.5	46081	46047
A-C	2.106	20.0		45700	0.056	88.0	9.4	44650	44634
Mean	1.555	26.3	1.2	217654	0.063	86.0	10.1	216181	216100

Table 5. For group 4 of the test instances from Tempelmeier and Derstroff (1996), Stadtler (2003) we compare in this table the results of Tempelmeier and Derstroff (1996) (T&D), Stadtler (2003) (Stadtler), and our results with 20 (ASMLCLS-20) and 30 minutes (ASMLCLS-30) computational time. MAPD indicates the mean deviation of the method in per cent to the best solution. % is the percentage of best solutions found. The last column shows the best solutions available.

Problem	T&D			Stadtler			Minimal
	MAPD	%	Cost	MAPD	%	Cost	cost
NC-1	2.65	33	364186	0.33	46.67	354709	353401
NC-2	6.75	0	1864407	0.63	20.00	1756314	1745334
NC-3	8.10	0	4035240	0.49	20.00	3750212	3731518
NC-4	5.52	0	2565054	0.68	26.67	2446366	2428613
NC-5	12.24	0	1838861	0.55	26.67	1648295	1639276
C-1	6.51	0	388112	0.08	40.00	359723	359432
C-2	8.70	0	2056183	1.47	13.33	1920455	1892262
C-3	7.84	0	4550056	1.24	0.00	4265387	4211022
C-4	9.91	0	2830900	2.70	6.67	2643123	2573430
C-5	10.14	0	1914168	2.40	6.67	1778768	1736597
Mean	7.83	3	2240717	1.06	20.67	2092335	2067089

Problem	ASMLCLS-20			ASMLCLS-30			Minimal
	MAPD	%	Cost	MAPD	%	Cost	cost
NC-1	0.28	53.33	354546	0.00	100.00	353401	353401
NC-2	0.67	6.67	1756868	0.02	86.67	1745710	1745334
NC-3	0.41	40.00	3747005	0.00	86.67	3731525	3731518
NC-4	0.33	46.67	2437987	0.14	93.33	2431617	2428613
NC-5	0.32	20.00	1644492	0.18	73.33	1642021	1639276
C-1	0.01	46.67	359473	0.00	93.33	359441	359432
C-2	0.25	40.00	1897090	0.01	93.33	1892534	1892262
C-3	0.66	0.00	4238226	0.00	100.00	4211022	4211022
C-4	0.96	13.33	2597734	0.001	93.33	2573459	2573430
C-5	0.52	40.00	1745506	0.00	100.00	1736597	1736597
Mean	0.44	30.67	2077893	0.04	92.00	2067733	2067089