A Hybrid Optimization Approach for Multi-Level Capacitated Lot-Sizing Problems

Christian Almeder

University of Vienna, Department of Business Administration
Bruenner Strasse 72, 1210 Vienna, Austria
christian.almeder@univie.ac.at

Abstract

Solving multi-level capacitated lot-sizing problems is still a challenging task, despite of increasing computational power and faster algorithms. In this paper a new approach combining an ant-based algorithm with an exact solver for (mixed-integer) linear programs is presented. A MAX-MIN ant system is developed to determine the principle production decisions, a LP/MIP solver is used to calculate the corresponding production quantities and inventory levels. Two different local search methods and an improvement strategy based on reduced mixed-integer problems are developed and integrated into the ant algorithm. This hybrid approach provides superior results for small and medium-sized problems in comparison to the existing approaches in the literature. For large-scale problems the performance of this method is among the best.

Keywords: Manufacturing, ant colony optimization, material requirements planning, mixed-integer programming.

1 Introduction

In this paper a general solution method for multi-level capacitated lot-sizing and scheduling problems (MLCLSP) is developed. This method combines
metaheuristics and exact LP/MIP solvers to determine production schedules as part of the material requirements planning (MRP) process. Determining the production schedule means to calculate the necessary amounts of raw materials, intermediate items and final products that should be purchased or produced in each period while minimizing total costs respecting the capacity limitations. In the production planning process the right decisions in lot-sizing have an essential impact on the system performance and its productivity and so also on the firm’s ability to remain competitive in the market. Hence, the development and the improvement of solution procedures for lot-sizing problems are very important.

The MLCLSP has been proven to be NP-hard by Maes et al. (1991). Nowadays commercial LP/MIP-Solvers, like CPLEX (© ILOG), or XpressMP (© Dash Optimizations), together with the increased computer power, are capable of solving larger and larger problems. But also real world applications are increasing in size, because of new and more complex production systems. Hence, those solvers are far away from solving general real world applications of the MLCLSP within an acceptable time span. Nevertheless those solvers have a good performance in solving large linear problems and problems with only a few binary decisions. For our approach we use this capability and combine it with metaheuristic methods which are known to be very efficient for combinatorial optimization problems.

In this study a hybrid algorithm is developed which is based on Ant Colony Optimization (ACO) used to select the principle production decisions, i.e. for which period production for an item should be scheduled. For that purpose a special version of ACO, namely a MAX-MIN Ant System, developed by Stützle and Hoos (1997) is used. Based on these production decisions selected by the ACO part the production amounts are determined by solving the remaining linear program using ILOG CPLEX. As an additional improvement step a reduced mixed-integer model is solved where most of the binary decisions are fixed. This approach works well for small and medium-size problems. In many cases this new algorithm improves the best solutions. For large instances the solution method leads to high-quality results, but cannot beat highly specialized algorithms.
Drexl and Kimms (1997) give a detailed review of the definition of different lot-sizing problems from simple single-level uncapacitated to complex multi-level capacitated problems. In particular, for the MLCLSP Maes et al. (1991) propose several LP-based heuristics restricted 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 successor and predecessor, and multiple resource constraints are allowed. Tempelmeier and Derstroff (1996) applied a Lagrangian 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 Barbarosoglu (2000) decompose the global problem (MLCLSP) into smaller subproblems and the intensive search capability of the simulated annealing is incorporated into the relaxation design. Stadler (2003) solves the MLCLSP by reducing the number of periods. He solves the problems on a rolling basis by adding later periods and removing earlier ones.

Several papers are dealing with metaheuristic approaches for solving the MLCLSP. The solution method of Xie and Dong (2002) is based on a similar idea, namely using the metaheuristic for determining the principal production decisions. They propose a genetic algorithm (GA) for selecting the production decisions combined with a simple heuristic to determine production quantities. Berretta and Rodrigues (2004) developed a memetic algorithm for the problem at hand and Berretta et al. (2005) describe metaheuristic methods to solve multi-level capacitated lot-sizing problems with non-zero lead times.

There are two papers about ant systems for a lot-sizing problem. Pitakaso et al. (2007) uses a combination of an ant system and a Wagner-Within model to solve uncapacitated problems item-per-item. Pitakaso et al. (2006) presents a very successful combination of a MAX-MIN ant system and an exact method. The ant system is used to decompose the problem into smaller subproblems which are solved exactly. This decomposition step for the MLCLSP is very complex and hard to apply to other variants of the MLCLSP. An extensive literature review on existing solution approaches for
lot-sizing problems can be found in Jans and Degraeve (2007).

The contribution of the present work can be summarized as follows:

- A successful and easy hybridization of exact and metaheuristic optimization approaches are developed. This is done in a very general way which can be applied to many other problems.

- A new and efficient algorithm for the MLCLSP is constructed and many new best solutions for well known test instances are obtained.

The paper is organized as follows. In Section 2 the mathematical formulation of the MLCLSP is given. In Section 3 the ACO algorithm is described, followed by an explanation of the local search methods in Section 4. Further improvements by a partial fixing method are are discussed in Section 5. Results are presented in Section 6 and Section 7 concludes with a summary and outlook for possible future research.

2 Mathematical formulation

In the case of the MLCLSP it is necessary to make production decisions for final products as well as for their components. There are several constraints and objectives that should be considered while developing a production schedule (cf. Drexl and Kimms, 1997):

- The given demand of the final products has to be fulfilled. No backorders or shortages are allowed.

- Each production decision incurs resource needs for setting up production (independent of the amount produced) and for the production of each item. The available resources are limited.

- Each production step needs a deterministic amount of time (measured in periods).

- Costs occur for setup and holding.
- Variable production costs are time-invariant. Therefore they can be ignored.

- Overtime is allowed to extend the available capacity, but it is penalized with high cost.

The mathematical formulation for the MLCLSP used in this study is based on the inventory & lot-size formulation by Stadtler (1996). The same formulation was used by Pitakaso et al. (2006).

Dimensions and indices:
- $P$ number of items 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:
- $S_i$ set of immediate successors of item $i$
- $s_i$ setup cost for item $i$
- $c_{ij}$ quantity of item $i$ required to produce one unit of item $j$
- $h_i$ holding cost for item $i$
- $a_{mi}$ capacity needed on resource $m$ for one unit of item $i$
- $b_{mi}$ setup time for item $i$ on resource $m$
- $L_{mt}$ available capacity of resource $m$ in period $t$
- $l_i$ lead time for production of item $i$
- $c_{om}$ overtime cost of resource $m$
- $G$ sufficiently large number
- $E_{it}$ external demand for item $i$ in period $t$
- $I_{i0}$ initial inventory of item $i$

Decision variables:
- $x_{it}$ delivered quantity of item $i$ at the beginning of period $t$
- $I_{it}$ inventory level of item $i$ at the end of period $t$
- $O_{mt}$ overtime hours used on resource $m$ in period $t$
binary variable indicating if 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:

\[
\min \sum_{i=1}^{P} \sum_{t=1}^{T} (s_{i}y_{i,t} + h_{i}I_{it}) + \sum_{t=1}^{T} \sum_{m=1}^{M} c_{m}^{0}O_{mt}
\]

subject to the set of constraints

\[
I_{it} = I_{it-1} + x_{it} - \sum_{j \in S_{i}} c_{ij}x_{jt} - E_{it} \quad \forall i, t
\]

\[
\sum_{i=1}^{P} (a_{mi}x_{it} + b_{mi}y_{it}) \leq L_{mt} + O_{mt} \quad \forall m, t
\]

\[
x_{it} - Gy_{it} \leq 0 \quad \forall i, t
\]

\[
I_{it} \geq 0, O_{mt} \geq 0, x_{it} \geq 0, y_{it} \in \{0, 1\} \quad \forall i, t
\]

The objective function (1) minimizes the sum of setup costs and inventory costs for all the items over a predefined planning horizon of length \(T\). Equations (2) represent the inventory balance equations. Capacity constraints (3) ensure that the resources required to produce the necessary amount of item \(i\) in period \(t\) plus the setup time does not exceed the available capacity. Constraints (4) capture 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 (5).

### 3 ACO Algorithm

First, the design of a MAX-MIN ant system algorithm is described for selecting the production periods for each item, i.e. the values of the variables
$y_{it}$ are determined by a metaheuristic. The proposed MAX-MIN ant system (MMAS) algorithm was developed by Stützle and Hoos (1997, 2000).

The basic idea of ACO is that a population of agents (artificial ants) repeatedly constructs solutions to a given instance of a combinatorial optimization problem (see Colomi et al., 1992; Dorigo and Stützle, 2004, for a detailed description of the principles of ACO). This construction phase is guided by a global memory (so called pheromone) holding information about parts of a solution, which have led to good results in previous generations of ants. After the construction phase of each ant a local search may be applied to improve the generated solutions. The pheromone values are updated based on the solution quality of the ants. The update is biased towards the best solutions found. A convergence proof for the general ACO algorithm can be found in Gutjahr (2003) and Stützle and Dorigo (2002).

For the considered lot-sizing problem each ant selects values (0 or 1) for the variables $y_{it}$. It starts from the top items down to the raw materials according to the ordering given by the bill of materials. The ant’s decision for production in a certain period is based on the pheromone information as well as on the heuristic information if there is an external (primary) or internal (secondary) demand (cf. equation (6)). The pheromone information represents the impact of a certain production decision on the objective values of previously generated solutions, i.e. the pheromone value is high if a certain production decision has lead to good solution in previous iterations.

On a subset of the test instances described in Section 6 different heuristic information (combinations of setup costs and holding costs) used by the ant for the construction of a solution and also the use of no heuristic information have been tested. It turned out that the best results can be reached by using the external and internal demand only. After the selection of the production decisions, we use a standard LP solver to solve the remaining linear problem. ($y_{it}$ is now a parameter and not a decision variable.) It may happen that a production lot, which was scheduled by the ant algorithm, is not used in the solution of the linear problem (i.e. $x_{it} = 0$ and $y_{it} = 1$). In such a case the production lot is removed (i.e. $y_{it}$ is set to 0) and the setup costs are corrected. Afterwards the solution is improved by a local search.
Procedure HybAntExact

    /* Initialization phase */
    Generate initial solution
    Initialize pheromone information
    while (termination condition not met) do
        for each ant do
            /* Construction phase */
            Construct a production scheme according to the decision rule (6).
            Solve the linear problem using a LP solver
            /* Local search method */
            Apply local search methods
        end
        /* Pheromone update phase */
        Update the pheromone information according to (9)
    end

Figure 1: Pseudo code of HybAntExact

method which is explained in the next section. After all ants of an iteration
have constructed a solution, the pheromone information is updated by the
iteration best as well as the global best solutions (using different weights).

In order to guide the search of the ants at the beginning, an initial feasible
solution is calculated using a simple heuristic: All binary variables \(y_{it}\) are set
to 1 and the according linear program is solved. Then the unused lots are
removed (i.e. \(y_{it}\) are set to 0 if the according \(x_{it} = 0\)) and the solution is
improved by a local search method. This solution is used for updating the
pheromone information as long as no better solution has been found by the
ants.

The pseudo code in Figure 1 illustrates the adaptations of the MMAS
to solve the MLCLSP. The ant system for multi-level capacitated lot-sizing
problems is denoted as HybAntExact in the sequel.

For the pheromone information representing the impact of selecting a
specific \(y_{it}\) to be zero or one, we use two different values, one for the 0-
decision (denoted by \(\tau_{it}^{(0)}(\ell)\)) and one for the 1-decision (denoted by \(\tau_{it}^{(1)}(\ell)\)).
The relation between these two values describes the probability of setting a
production decision or not (i.e. setting \(y_{it}\) to zero or to one). This information
is used in the construction phase of the algorithm.
Remark. A single pheromone value between 0 and 1 would be enough to encode that information. But because of an easier and faster implementation, separate pheromone values for the 0- and 1-decisions are used.

In the construction phase an ant decides independently item-per-item about the production periods. The random proportional rule (6) is used to determine the probability that an item is selected for production in a certain period (cf. Dorigo and Stützle, 2004).

\[ p_{i\kappa}^\kappa(\ell) = \frac{\left[\tau_{it}^{(1)}(\ell)\right]^\alpha \left[\eta_{it}^{(1)}(\ell, \kappa)\right]^\beta}{\left[\tau_{it}^{(0)}(\ell)\right]^\alpha \left[\eta_{it}^{(0)}(\ell, \kappa)\right]^\beta + \left[\tau_{it}^{(1)}(\ell)\right]^\alpha \left[\eta_{it}^{(1)}(\ell, \kappa)\right]^\beta} \]  (6)

As a result of tests on a subset of instances the following parameters for the heuristic information based on the external and internal demand are used:

\[ \eta_{it}^{(1)}(\ell, \kappa) = \begin{cases} 1.8 & \text{if there is a demand for item } i \text{ in period } t, \\ 1 & \text{otherwise}, \end{cases} \]  (7)

\[ \eta_{it}^{(0)}(\ell, \kappa) = 2.8 - \eta_{it}^{(1)}(\ell, \kappa), \]  (8)

The ant determines for each period-product combination \( i-t \) the value of the binary variable \( y_{it} \), starting with the end items. Based on the ant’s selection for the end items the heuristic information for the predecessors are calculated and the ant continues with the direct predecessors until it reaches
the raw materials. In order to avoid infeasible solutions, the ant’s selection is corrected if it does not schedule production before the first demand occurs. In that case production is scheduled at a single random period before the first (external or internal) demand. After that construction phase, the reduced linear program — given by (1)-(5) and fixing the predetermined variables $y_{it}$ — is solved to optimality using the LP solver. The final solution is improved by applying a local search method described in the next section.

The update of the pheromone information consists of 2 parts: (i) the information evaporates (i.e. the pheromone values are diminished by a factor $\rho$), (ii) the pheromone values for decisions which are part of the global and the iteration best solutions are increased. Several tests have shown that the iteration best as well as the global best solution should update the pheromone information. Typically for a MAX-MIN ant system the pheromone values are bounded to an interval $[\tau_{\min}, \tau_{\max}]$; in this case $[0.01, 1.0]$:

\[
\rho \in [0, 1] \quad \text{trail persistence parameter to regulate the evaporation of } \tau_{pj}
\]

\[
\Delta \tau_{it}^{(0/1)}(\ell) \quad \text{total increase of pheromone on edge } (i, t) \text{ for } \tau_{it}^{(0)} \text{ respectively } \tau_{it}^{(1)}.
\]

\[
f(s^{\text{opt}}) \quad \text{global best objective value}
\]

\[
f(s^{it}) \quad \text{iteration best objective value}
\]

\[
\tau_{it}^{(0/1)}(\ell + 1) = \max \left( \tau_{\min}, \min \left( \tau_{\max}, \rho \tau_{it}^{(0/1)}(\ell) + \Delta \tau_{it}^{(0/1)}(\ell) \right) \right). \quad (9)
\]

with

\[
\Delta \tau_{it}^{(0/1)}(\ell) = \begin{cases} 
\frac{c_1}{f(s^{\text{opt}})} + \frac{c_2}{f(s^{it})} & y_{it} = 0/1 \text{ for the global and iteration best ant} \\
\frac{c_1}{f(s^{\text{opt}})} & y_{it} = 0/1 \text{ for the global best ant} \\
\frac{c_2}{f(s^{it})} & y_{it} = 0/1 \text{ for the iteration best ant}
\end{cases}
\]

4 Local search methods

In order to improve the solutions constructed by the ants two different local search procedures are developed. The first one is a fast search method based
on moving parts of the production from one period to another one. It will be denoted as \textit{quantity-based local search}. The second one is based on the idea of keeping some of the production decision made by the ant as fixed, and others are free. The resulting MIP is solved with a LP/MIP solver. The second method will be called \textit{setup-based local search}. The results described in Section 6 show that the \textit{quantity-based} method is faster and leads to considerable good results. The \textit{setup-based} method is more time-consuming and delivers slightly worse results. Nevertheless a sequential combination of both methods, meaning to apply one after the other, can improve the solution quality even further.

4.1 Quantity-based local search

The basic operation for this neighborhood search is the \textit{move}-operation. It moves a part or the whole production of a certain item from one period to another one. In order to keep feasibility with respect to the inventory balance equations (2) when moving production to previous periods, a move may lead to additional moves for predecessor items.

The whole local search procedure consists of three steps:

1. Reduce overtime by moving production backwards from periods with overtime to the nearest period with free capacity.

2. Reduce setup by moving the entire production of an item of one period to a previous production period.

3. Reduce inventory by moving production forward in time towards the demand period.

All three steps are applied iteratively until no improvement is possible. Since the move operations do not preserve the optimality property of the solution regarding the reduced linear program, the reduced linear program is solved after each step and at the end of the local search using the production decisions resulting from the neighborhood search. Figure 2 shows the pseudo code for this local search method.
**Procedure** Quantity-based local search

```plaintext
do
  Randomly choose a period-resource combination \((t, m)\) with \(O_{mt} > 0\)
do
    for each item (random sequence) using resource \(m\) in period \(t\) do
      Try to move production backwards to a period with free capacity.
    end for
  while another period-resource combination with \(O_{mt} > 0\) available
Solve the linear problem using the LP/MIP solver.
Randomly choose a period-item combination \((t, i)\) with \(x_{it} > 0\)
do
  if \(I_{it} \geq x_{it}\)
    Try to move production forward.
  else
    Try to move production backwards.
  end if
while another period-item combination with \(x_{it} > 0\) available
Solve the linear problem using the LP/MIP solver.
Randomly choose a period-item combination \((t, i)\) with \(I_{it} > 0\)
do
  Try to move production forward as far as possible.
while another period-item combination with \(I_{it} > 0\) available
Solve the linear problem using the LP/MIP solver
while solution has been improved
```

Figure 2: Pseudo code of quantity-based local search
4.2 Setup-based local search

The second local search method is based on the mixed-integer formulation of the MLCLSP given in (1)-(5). The main idea is to take the solution generated by the ant and recalculate some of the binary decisions $y_{it}$, i.e. some of these decisions are assumed to be fixed, others are released. Hence, the result is a mixed-integer program as for the full MLCLSP, but with most of the binary decisions already set. This reduced MIP is solved using a LP/MIP solver.

A crucial part of the method is to determine which and how many binary decisions should be released. The number of free decisions should be small enough to guarantee that the resultant MIP can be solved within a very short computational time (or at least that a good solution can be found). But increasing the number of free decisions may lead to better solutions because of the increased search space. In general, a feasible solution is already known before solving the MIP. Therefore, a larger amount of free decisions is possible. The considered test instances in Section 6 consist of up to 1600 binary decisions. Several tests have shown that a threshold of 10 percent of free decisions seems a good choice for medium and large problems, because at least an improvement of the solution can be reached within short computational time. Nevertheless it may be necessary to decrease this threshold for significantly larger problems.

The selection of these free decision variables is based on a similar idea as the quantity-based local search procedure. That means that some of these free decisions are selected to possibly reduce overtime, some are selected to reduce setup, and others are selected to reduce inventory levels. So instead of actual performing the move operation as in the quantity-based local search, the binary decisions $y_{it}$ for the according item and period are released, the remaining ones are fixed. Only a third of the total allowed free decisions are dedicated to each of the goals: overtime, setup, and inventory.

The computational time for solving the resulting MIP with the LP/MIP solver is limited. Since the MIP will be solved several times for each ant, it is necessary to keep the time low. Definitely, it is necessary to adapt this time to the problem size. For the test instances the time limits were set between
**Procedure** Setup-based local search

```
  do
    Fix all binary decisions according to the current solution.
    Release 3.3% of the binary decisions to reduce overtime.
    Release 3.3% of the binary decisions to reduce setup.
    Release 3.3% of the binary decisions to reduce inventory.
    Start LP/MIP solver and try to improve the solution.
  while solution can be improved and iterations < 2
```

Figure 3: Pseudo code of quantity-based local search

0.1 seconds for small instances and 0.5 seconds for large instances. In most cases the LP/MIP solver can find improved solution within this time span.

Although the calculation time for the MIP is restricted it has a considerable impact on the overall computational time. Therefore this search method is restricted to two iterations, allowing only a second round if the first step has succeeded and improved the solution quality. The overall pseudo code for this search method is depicted in Figure 3.

## 5 Pheromone-based partial fixing

Based on the idea of the *setup-based local search* described in Section 4.2 a technique to improve the intensification phase of the MAX-MIN ant system was developed. The pheromone values of the ant system provide information about the selection of production decisions. For some decisions the pheromone values indicate clearly to set a lot or not to set a lot, for others the pheromone values may deliver an ambiguous picture. So it may be possible to speed up the intensification phase of the ant system by using this information in constructing a reduced MIP similar to the previous method. The pheromone information is analyzed and those production decisions $y_{it}$ where the pheromone information indicates clearly that 0 or 1 is the better choice are fixed. Other production decisions, where the pheromone values do not clearly show what a good selection might be, are kept as free decisions. In order to avoid too many free decisions the number of free variables is reduced.
randomly if necessary. Analogously, the number of free variables is increased randomly, if the pheromone values lead only to a few free production decisions. The reduced MIP is solved, but in that case the allowed calculation time is set to several seconds. This improvement step is performed regularly after every 100 iterations as well as at the end of the algorithm.

6 Results

The following variations of the HybAntExact algorithm were tested:

- **HybAntLP**: MAX-MIN ant system with quantity-based local search.
- **HybAntMIP**: MAX-MIN ant system with setup-based local search.
- **HybAntLPMIP**: MAX-MIN ant system with quantity-based (first) and setup-based (second) local search.
- **HybAntMIPLP**: MAX-MIN ant system with setup-based (first) and quantity-based (second) local search.
- **HybAntLPFix**: MAX-MIN ant system with quantity-based local search and pheromone-based partial fixing.
- **HybAntMIPFix**: MAX-MIN ant system with setup-based local search and pheromone-based partial fixing.
- **HybAntMIPLPFix**: MAX-MIN ant system with setup-based (first) and quantity-based (second) local search and pheromone-based partial fixing.

The algorithms were implemented in C++ (gcc 4.1) using CPLEX 10.1 for solving the linear and mixed-integer problems. The tests were performed on a personal computer *Pentium D 3.2GHz* with 4 GB RAM and *SUSE Linux 10.1*.

The algorithms were tested using different groups of test instances taken from Tempelmeier and Derstroff (1996) and Stadtler (2003) and comparing...
the results with those of Pitakaso et al. (2006), which are the best results published so far. For all further test the following parameter settings which where a result based on pretest on a subset of test instances have been used:

- **Number of ants per iteration:** 5
  Tests have been performed using 5 and 10 ants per iteration. The difference of the solution quality turned out to be negligible, whereas the computational time when using 5 ants was almost 50 percent lower.

- **Minimum and maximum pheromone value:** 0.01, 1.0
  The possible decisions of an ant during the construction phase does not depend on previous decisions of the ant. Hence, it is not necessary to use adaptive pheromone bounds as suggested by Stützle and Hoos (2000).

- **Initial pheromone values:** \([0.8, 1.0]\) - depending on the total load of the according resource.
  Tests with constant and adaptive initial pheromone values have been performed. The above values caused a speed up of convergence without decreasing the solution quality.

- **Evaporation factor:** 0.97
  Evaporation factors of 0.95, 0.97, and 0.98 have been tested. The factor of 0.97 gives the best trade-off between fast convergence and good solution quality.

- **Stopping criteria:** 150 iterations without improvement and time limit (depending on test instance).
  Tests have shown that after 100-150 iterations without improvement the convergence slows down significantly and only small improvements of the solution are possible with a high computational effort.

Group 1 consists of 600 test instances with 10 items, 4 periods, and 3 resources.\(^1\) There are instances with general systems (G) and instances with assembly systems (A) and two different ways of assigning the resources.

\(^1\)This group of test instances is group B from Tempelmeier and Derstroff (1996).
In the non-cyclic case (NC), for each production level just one 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.

The test examples are small enough to calculate the optimal solution within a few seconds using CPLEX or XPRESS-MP. Only the HybAntLP algorithm has been applied to check the performance of the newly developed method without partial fixing or setup-based local search. Due to the small instances the partial fixing step would finally calculate the full MIP and therefore deliver the optimal solution. For the proposed algorithm we apply a time limit of 10 minutes, but all instances stop after a few seconds because of the second criteria of 150 non-improving iterations.

Table 1 shows a comparison of the results of the different methods. With HybAntLP it is possible to find nearly all optimal solutions and the gap for the remaining ones is small. The ASMLCLS delivers also very good results for that small test instances. The reason for such small deviations from the optimal solutions is based on the algorithm’s decomposition method into subproblems. The maximum size of those subproblems is nearly as large as the whole problem itself. So basically the ASMLCLS solves almost the whole problem exactly in one step.

Tempelmeier and Derstroff did their test in 1996 on a computer which is more than 1000 times slower than the computer used for the current tests. This means their method is extremely fast, but they reported in the paper that it was not possible to improve the results significantly if they would have increased the number of iterations.

To allow a more extensive testing a group of slightly larger test instances was used. It is based on the previous group, but the planning horizon includes 24 instead of 4 periods. Although the size of the problems is small, it is not possible to obtain the optimal solution within several hours or days (using the computer equipment mentioned above). The group consists of 120 instances, both general and assembly structure. Stadtler (2003) reports an average deviation of 1.06% from the best known solutions within 120s. of

---

2 This group of test instances is group A+ from Stadtler (2003).
Table 1: Comparison of the results of different methods for group 1 of the test instances from Tempelmeier and Derstroff (1996). HybAntLP is compared with T&D (Tempelmeier and Derstroff, 1996) and ASMLCLS (Pitakaso et al., 2006). MAPD indicates the mean percentage deviation of the method from the optimal solution. The rows best sol. show the percentage of optimal solutions found.

<table>
<thead>
<tr>
<th>Structure</th>
<th>G-NC</th>
<th>G-C</th>
<th>A-NC</th>
<th>A-C</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>T &amp; D</td>
<td>MAPD</td>
<td>1.60%</td>
<td>2.18%</td>
<td>0.64%</td>
<td>0.79%</td>
</tr>
<tr>
<td></td>
<td>best sol.</td>
<td>37.3%</td>
<td>25.3%</td>
<td>64.7%</td>
<td>45.3%</td>
</tr>
<tr>
<td>ASMLCLS</td>
<td>MAPD</td>
<td>0.005%</td>
<td>0.017%</td>
<td>0.006%</td>
<td>0.002%</td>
</tr>
<tr>
<td></td>
<td>best sol.</td>
<td>93.3%</td>
<td>88.7%</td>
<td>92.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>HybAntLP</td>
<td>MAPD</td>
<td>0.089%</td>
<td>0.086%</td>
<td>0.459%</td>
<td>0.013%</td>
</tr>
<tr>
<td></td>
<td>best sol.</td>
<td>94.0%</td>
<td>89.3%</td>
<td>96.7%</td>
<td>93.3%</td>
</tr>
</tbody>
</table>

computational time. Since the HybAntExact algorithm can improve 58.3% of these best known solutions, the new average gap for the results of Stadtler (2003) would be 1.64%.

Table 2 shows an overview for the results of the different version of our algorithm. HybAntLP and HybAntLPfix deliver good results in a short time. But it is not possible to improve the solutions substantially by allowing longer calculation times. Furthermore, it shows that the pheromone-based partial fixing improves the solution quality while keeping the same computational time. HybAntMIP and HybAntMIPfix use considerable more computational time, because the setup-based local search is more expensive. Both methods are not able to find for all test instances capacity feasible solutions, although such solutions are known. In the case of HybAntMIP 7 instances and in the case of HybAntMIPFix 3 instances lead to solutions which exceed the available capacity. If those instances are removed, the mean average deviation is low. The next 2 blocks in the table show the results, if both local search methods are combined. The results are significantly better, if the setup-based local search is applied first, and afterwards the quantity-based

---

3The results reported by Tempelmeier and Derstroff (1996) and Pitakaso et al. (2006) for the A-C group consider by mistake only 75 instead of 150 instances. Each of these 75 instances was solved twice. In order to compare the HybAntLP method with the other algorithms, only the results of these 75 instances were considered.
Table 2: Comparison of the results of different methods for group 2 of the test instances from Stadtler (2003). MAPD indicates the mean deviation of the method in per cent to the best solution. Number in parentheses indicate MAPD without capacity infeasible solutions. The rows best sol. show the percentage of best solutions found.

<table>
<thead>
<tr>
<th>Structure</th>
<th>General</th>
<th>Assembly</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stadtler</td>
<td>MAPD</td>
<td>-</td>
<td>1.64%</td>
</tr>
<tr>
<td>(2.0 min.)</td>
<td>best sol.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HybAntLP</td>
<td>MAPD</td>
<td>2.78%</td>
<td>2.53%</td>
</tr>
<tr>
<td>(1.9 min.)</td>
<td>best sol.</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>HybAntMIP</td>
<td>MAPD</td>
<td>47.74% (3.72%)</td>
<td>52.75% (3.85%)</td>
</tr>
<tr>
<td>(9.5 min.)</td>
<td>best sol.</td>
<td>3.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>HybAntLPFix</td>
<td>MAPD</td>
<td>2.51%</td>
<td>2.23%</td>
</tr>
<tr>
<td>(1.9 min.)</td>
<td>best sol.</td>
<td>3.3%</td>
<td>3.3%</td>
</tr>
<tr>
<td>HybAntMIPFix</td>
<td>MAPD</td>
<td>15.50% (1.83%)</td>
<td>30.11% (2.15%)</td>
</tr>
<tr>
<td>(9.2 min.)</td>
<td>best sol.</td>
<td>5.0%</td>
<td>3.3%</td>
</tr>
<tr>
<td>HybAntLPMIP</td>
<td>MAPD</td>
<td>2.32% (2.32%)</td>
<td>3.94% (1.99%)</td>
</tr>
<tr>
<td>(6.8 min.)</td>
<td>best sol.</td>
<td>11.7%</td>
<td>10.0%</td>
</tr>
<tr>
<td>HybAntMIPLP</td>
<td>MAPD</td>
<td>1.28%</td>
<td>1.01%</td>
</tr>
<tr>
<td>(7.1 min.)</td>
<td>best sol.</td>
<td>11.7%</td>
<td>25.0%</td>
</tr>
<tr>
<td>HybAntMIPLPFix</td>
<td>MAPD</td>
<td>0.99%</td>
<td>0.86%</td>
</tr>
<tr>
<td>(6.9 min.)</td>
<td>best sol.</td>
<td>13.3%</td>
<td>28.3%</td>
</tr>
</tbody>
</table>

A Wilcoxon signed-rank test was applied to identify significant differences between the different versions of the algorithm. The results in Table 3 show that HybAntMIPLP and HybAntMIPLPFix perform best. Since the latter delivers slightly better results with less computational time, we will use mainly this version for further testing.

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. This group of test instances is group C from Tempelmeier and Derstroff (1996).
Table 3: This table shows the results of the Wilcoxon signed-rank test with \( \alpha = 0.99 \). > indicates that the algorithm in the column gives significant better results than the algorithm of the row. = expresses that the algorithms show now significant difference.

<table>
<thead>
<tr>
<th></th>
<th>HybAntLP</th>
<th>HybAntLPFix</th>
<th>HybAntMIP</th>
<th>HybAntMIPFix</th>
<th>HybAntLPMIP</th>
<th>HybAntMIPLP</th>
<th>HybAntMIPLPFix</th>
</tr>
</thead>
<tbody>
<tr>
<td>HybAntLP</td>
<td>&lt; &gt; = &gt; &gt; &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HybAntMIP</td>
<td>&gt; &gt; = &gt; &gt; &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HybAntLPFix</td>
<td>= &gt; &gt; &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HybAntMIPFix</td>
<td>&gt; &gt; &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HybAntLPMIP</td>
<td>&gt; &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HybAntMIPLP</td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HybAntLPMIPFix is superior to the methods of Tempelmeier and Derstroff (1996) and Pitakaso et al. (2006). For 86% of the instances our algorithm delivers best results. The runtime of the algorithm is comparable with those of the ASMLCLS considering the different computers used.

Finally, group 4 was investigated, consisting of 150 large test instances with 100 items spread over 10 levels, 16 periods, and 10 resources all with a general product structure.\(^5\) The results of the proposed algorithm are compared with those of Tempelmeier and Derstroff (1996); Stadtler (2003); Pitakaso et al. (2006). For these large test instances the method of Tempelmeier and Derstroff is extremely fast, but has very low solution quality. The method developed by Stadtler is very complex consuming a lot more computational time, but the results from Tempelmeier and Derstroff (1996) were improved by more than 5%. The best results were obtained by the ASMLCLS algorithm presented in Pitakaso et al. (2006). In Table 5 the detailed results are reported. Since detailed results are not available for all methods only the difference to the best solutions available is reported. Therefore also a negative MAPD is possible. The computational times reported in

\(^5\)This group of test instances is group E from Tempelmeier and Derstroff (1996).
Table 4: Comparison of the results of different methods for group 3 Tempelmeier and Derstroff (1996). The results of HybAntMIPLPFix are compared with those of T&D (Tempelmeier and Derstroff, 1996) and ASMLCLS (Pitakaso et al., 2006). MAPD indicates the mean deviation of the method in per cent to the best solution. The rows best sol. show the percentage of best solutions found.

<table>
<thead>
<tr>
<th>Structure</th>
<th>General Assembly</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>T &amp; D</td>
<td>MAPD 7.11%</td>
<td>6.18%</td>
</tr>
<tr>
<td></td>
<td>best sol. 8.3%</td>
<td>6.3%</td>
</tr>
<tr>
<td>ASMLCLS</td>
<td>MAPD 6.24%</td>
<td>3.99%</td>
</tr>
<tr>
<td></td>
<td>best sol. 9.7%</td>
<td>12.3%</td>
</tr>
<tr>
<td>HybAntMIPLPFix</td>
<td>MAPD 0.07%</td>
<td>0.27%</td>
</tr>
<tr>
<td></td>
<td>best sol. 97.3%</td>
<td>93.0%</td>
</tr>
</tbody>
</table>

Since the setup-based local search is very time consuming, especially for such large instances, the HybAntLPFix version was tested, too. Surprisingly, it delivers equal good results than the full HybAntMIPLPFix (using a Wilcoxon signed-rank test), but the computational time drops down by more than 25%. The solution quality obtained by these two versions is better than the method by Tempelmeier and Derstroff (1996) and is nearly as good as the results reported by Stadtler (2003), but cannot reach the best results produced by the ASMLCLS by Pitakaso et al. (2006).

7 Conclusions

In this paper a combination of a metaheuristic and an exact LP/MIP solver applied to multi-level capacitated lot-sizing problems have been analyzed. This straight-forward approach is characterized by determining binary decision by a MAX-MIN ant system, and computing the continuous variables using an exact approach. Furthermore 2 different local search methods have been developed, a fast one based on shifting production and solving a linear program, and a more complex one, where the capability of a MIP solver
Table 5: Comparison of the results of different methods for group 4 from Tempelmeier and Derstroff (1996); Stadtler (2003). The results of HybAntLPFix and HybAntMIPLPFix are compared with T&D (Tempelmeier and Derstroff, 1996), Stadtler (Stadtler, 2003), and ASMLCLS (Pitakaso et al., 2006). MAPD indicates the mean deviation of the method in per cent to the best solution available.

<table>
<thead>
<tr>
<th>Structure</th>
<th>General Assembly</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>T &amp; D (2 min.)</td>
<td>MAPD 6.56%</td>
<td>7.03% 6.79%</td>
</tr>
<tr>
<td>Stadtler (20 min.)</td>
<td>MAPD -</td>
<td>- 1.7%</td>
</tr>
<tr>
<td>ASMLCLS (30 min.)</td>
<td>MAPD -</td>
<td>- -1.0%</td>
</tr>
<tr>
<td>HybAntLPFix (28 min.)</td>
<td>MAPD 2.68%</td>
<td>1.26% 1.97%</td>
</tr>
<tr>
<td>HybAntMIPLPFix (39 min.)</td>
<td>MAPD 2.27%</td>
<td>1.77% 2.02%</td>
</tr>
</tbody>
</table>

was used to improve solutions for small mixed-integer problems. As a further improvement strategy a pheromone-based intensification step has been implemented, where the information gathered by the ant system is used to construct smaller mixed-integer problems.

The results reported in Section 6 have shown that this approach is superior for small and medium sized instances. It is possible to improve the best known results for those instances. For large instances, the solution is among the best. The conclusions that can be drawn from these results are that ant colony optimization is a very competitive method if the solution space is not extremely large. For very large problems an approach, where the ants are searching for a good decomposition of the problem, might be favorable, because the search space is reduced (cf. Pitakaso et al., 2006). This observation is consistent with results for other problems (cf. Reimann et al., 2004).

The presented solution approach is very general. The main idea of using a metaheuristic to select the binary decisions and then solve a much simpler problem exact can be easily adapted to modified lot-sizing problems or even to other planning problems. Also other population-based metaheuristic methods may be used in a similar way to solve such problems. But those aspects are beyond the scope of this paper and will be a subject for future research.
Acknowledgment

I want to thank Richard Hartl and Karl Doerner for several interesting discussions on ant systems and local search methods.

References


Tempelmeier, H., Derstroff, M., 1996. A langrangean-based heuristi for dy-
namic multilevel multi-item constrained lotsizing with setup times. Man-

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

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