Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 2533–2552
computers & operations research
www.elsevier.com/locate/cor

A MAX-MIN ant system for unconstrained multi-level lot-sizing problems

Rapeepan Pitakaso[a, b], Christian Almeder[b,*], Karl F. Doerner[b], Richard F. Hartl[b]

[a] *Department of Industrial Engineering, Ubonrajathanee University, Thailand 34190*
[b] *Institute for Management Science, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria*

Available online 6 June 2006

Abstract

In this paper, we present an ant-based algorithm for solving unconstrained multi-level lot-sizing problems called ant system for multi-level lot-sizing algorithm (ASMLLS). We apply a hybrid approach where we use ant colony optimization in order to find a good lot-sizing sequence, i.e. a sequence of the different items in the product structure in which we apply a modified Wagner–Whitin algorithm for each item separately. Based on the setup costs each ant generates a sequence of items. Afterwards a simple single-stage lot-sizing rule is applied with modified setup costs. This modification of the setup costs depends on the position of the item in the lot-sizing sequence, on the items which have been lot-sized before, and on two further parameters, which are tried to be improved by a systematic search. For small-sized problems ASMLLS is among the best algorithms, but for most medium- and large-sized problems it outperforms all other approaches regarding solution quality as well as computational time.

Keywords: Ant colony optimization; Multi-level lot-sizing; Wagner–Whitin algorithm; Material requirements planning

1. Introduction

Within computer-based enterprise resource planning (ERP) systems, such as SAP or BAAN, the part material requirements planning (MRP) generates a production plan for each item over a given planning horizon. The production plan of an end item causes secondary demand for its components. Taking into account purchasing, production, or transportation lead times as well as current and planned inventory levels, the MRP approach determines the lot-sizes of each component in all periods. Extending MRP, manufacturing resource planning (MRP II) also checks for capacity violations in a capacity requirements planning (CRP) module. Afterwards, jobs are released and scheduled on the machines.

The basic sequential MRP framework of master production scheduling, material requirements planning and lot-sizing, capacity requirements planning and finally scheduling is known to have several key weaknesses:

1. The treatment of capacities is usually done in a rather rudimentary way. Uncapacitated lot-sizing approaches are typically used to derive production plans whose capacity is evaluated through the CRP module. CRP simply provides

* Corresponding author.
E-mail addresses: rapheephan.p@ubu.ac.th (R. Pitakaso), christian.almeder@univie.ac.at (C. Almeder), karl.doerner@univie.ac.at (K.F. Doerner), richard.hartl@univie.ac.at (R.F. Hartl).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.09.022

a signal that the production plan violates capacity limitations. Thus, it can be considered as an information-gathering procedure rather than a decision-making one. Afterwards an iterative search for a schedule that satisfies the capacity constraints follows. However, material and capacity planning should be done in parallel, rather than in sequence. These capacitated multi-level lot-sizing (MLLS) problems have been studied in a series of papers, see e.g. the conceptual discussion by Billington et al. [1] or the heuristic approaches by Tempelmeier and Derstroff [2] or Tempelmeier and Helber [3].

2. Since lead times are in fact dependent on product mix, shop load, and capacity they should be viewed not as inputs to a scheduling procedure but rather as part of the output. Thus, it would be desirable to do some simultaneous planning over several of the steps mentioned, e.g. lot-sizing and scheduling (see e.g. [4]).
3. Even when disregarding the above points (1) and (2), MRP systems do not really provide an efficient methodology since commercially-available MRP softwares typically use the simplest (and suboptimal) lot-sizing approaches.

While (1) and (2) are important, this paper focuses on point (3). One could justify this by the fact that, in practice, uncapacitated lot-sizing models continue to be largely used since the implementation of capacitated approaches requires a lot of data which firms are often reluctant to collect or maintain. Also, we plan to extend the approach used here to capacitated problems.

In what follows, we consider uncapacitated lot-sizing problems. The single-level lot-sizing (SLLS) problem is the simplest category among these. Wagner and Whitin [5] introduced this model and developed a well-known exact algorithm based on dynamic programming. Subsequently a large number of heuristics was developed such as Silver and Meal [6] based on the idea of minimizing average setup cost and inventory cost over several periods.

Multi level lot-sizing (MLLS) problems is a more general class of combinatorial optimization problems, where the aim is to find a production plan for all items in the bill of materials (BOM). It should be noted that this problem class is more difficult than SLLS and is usually solved heuristically. There are three types of product structures: (a) assembly system (each item may have several predecessors, but at most one successor), (b) serial system (each item has at most one predecessor and one successor), and (c) the general system (each item can have more than one successor and predecessor). Several solution approaches have been developed for each of these three problem classes. We will consider the general systems which includes also the other two types.

The most straightforward way to solve MLLS problems is to use decomposition by product. First, lot-sizing is done for all end items using a SLLS technique. With the secondary demand induced by lots of direct successors, the components of the highest level in the BOM are lot-sized, and so on until the raw materials are reached; see e.g. [7,8]. It has been known for a long time that this approach disregards the effects of lot-sizing on the parts contained in the current product which leads to solutions which induce unnecessary high setup and holding costs.

Modifying costs is a way to achieve some degree of inter-level coordination. Setup costs are altered to account for the fact that placing an order for a certain item can trigger orders for its predecessors. Some approaches also modify the holding costs. These adjusted costs are used when applying some SLLS method to each item in the product structure. Blackburn and Millen [9] designed five types of adjustment for setup and holding cost and used them in single end-item assembly systems. Bookbinder and Koch [10] extended this approach to systems with a general product structure.

Recently, Dellaert and Jeunet [11] proposed a randomized MLLS heuristic for general product structures. Their contribution was twofold: (1) instead of using a constant modified setup cost for the whole planning horizon, they allowed time-varying setup cost. In each period, the setup costs of each product are only adjusted by those predecessors without positive demand. If a predecessor faces some primary (external) demand or has to be produced because of secondary (internal) demand caused by some other product, adding some secondary demand would not cause any setup. Thus, the setup costs of this predecessor should be disregarded in the adjustment process of the setup costs. (2) They also randomized their algorithm by multiplying the adjustment term with a certain factor. The "optimal" value of this factor was determined using Monte Carlo simulation.

A number of metaheuristics have also been applied to MLLS problems. Kuik and Salomon [12] introduced a simulated annealing algorithm, Dellaert and Jeunet [13] developed a genetic algorithm, and Jeunet and Jonard [14] proposed a new heuristic based on three single-point stochastic searches: simulated annealing, simulated tempering, and hill climbing. These recent approaches will serve as benchmarks for our algorithm which is based on ant colony optimization (ACO).

The ACO metaheuristic was first proposed by Colorni et al. [15] in 1992. The main idea of ACO is that a population of artificial ants repeatedly builds and improves solutions to a given instance of a combinatorial optimization problem.

From one generation to the next a global memory is updated that guides the construction of solutions in the successive population. The best solutions found so far by the ants are used for the memory update. After the construction phase of the algorithm usually a local search is applied to improve the solutions of the ants.

ACO algorithms have been successfully applied to a variety of combinatorial optimization problems such as the traveling salesperson problem, the quadratic assignment problem, different variants of the vehicle routing problem, the graph colouring problem and different variants of machine scheduling problems. For an overview of the most successful applications we refer to Dorigo and Stützle [16]. A convergence proof for an ACO algorithm can be found in [17,18].

Our ant system for multi-level lot-sizing (ASMLLS) algorithm is based on the MAX-MIN ant system (MMAS), proposed by Stützle and Hoos [19,20], which is a particular variant of the ACO metaheuristics. A convergence proof for the MMAS can be found in [18].

The contribution in this paper is threefold:

- While previous metaheuristics approaches to MLLS problems (see above) always worked directly using the IP-formulation of the problem, we demonstrate that an effective way of quickly obtaining new best solutions is to use the constructive randomized MLLS heuristic of Dellaert and Jeunet [11] and to optimize the lot-sizing sequence of the products using ACO.
- We show that it is advantageous not to keep constant the adjustment factor in the modified setup cost approach over the construction phase but to choose different factors for the early products (end items) and the late products lot-sized (parts).
- In fact, we optimize both, the average value of this adjustment factor and the increase or decrease of this factor using an evolutionary approach in connection with the ACO algorithm. In this sense our algorithm could be viewed as being hybrid.

Using all available test instances from the literature we show that all of the above design decisions of our approach contribute to the solution quality and that for medium- and large-size benchmark instances our algorithm outperforms all other algorithms published so far.

The paper is organized as follows: Section 2 is dedicated to the presentation and mathematical formulation of the MLLS problem. In Section 3, a general framework of the proposed algorithm will be discussed. The experimental framework and the computational results will be presented in Section 4. Finally, conclusion and outlook can be found in Section 5.

2. Model formulation

In this section we formulate a mixed-integer program (MIP) which represents the MLLS problem. First, we introduce the necessary notation:

$\Gamma(i)$	set of immediate successors of item i
$\Gamma^{-1}(i)$	set of immediate predecessors of item i
$c_{i,j}$	quantity of item i required to produce one unit of item j
$D_{i,t}$	external requirement for item i in period t
h_i	holding cost for item i (assumed to be constant over time)
$I_{i,0}$	initial inventory of product i
l_i	lead time of item i
P	total number of items (end-products, sub-assemblies and raw materials)
s_i	setup cost for item i (assumed to be sequence-independent)
T	total number of periods

For the decision and auxiliary variables we use the following notation:

$d_{i,t}$	total requirement for item i in period t
$I_{i,t}$	inventory level of item i at the end of period t
$x_{i,t}$	delivered quantity of item i at the beginning of period t
$y_{i,t}$	binary variable which indicates if an item i is produced in period t ($y_{i,t} = 1$) or not ($y_{i,t} = 0$)

The resulting MIP can be written as follows:

$$\min_{y_{i,t},x_{i,t},I_{i,t},d_{i,t}} \sum_{i=1}^{P}\sum_{t=1}^{T}(s_i y_{i,t} + h_i I_{i,t}) \tag{1}$$

subject to (each constraint must hold $\forall i, t$)

$$I_{i,t} = I_{i,t-1} + x_{i,t} - d_{i,t}, \tag{2}$$

$$d_{i,t} = \begin{cases} D_{i,t} & \text{if } i \text{ is an end-item } (\Gamma(i) = \emptyset), \\ \sum_{j\in\Gamma(i)} c_{i,t}x_{j,t+l_j} + D_{i,t} & \text{otherwise } (\Gamma(i) \neq \emptyset), \end{cases} \tag{3}$$

$$x_{i,t} - My_{i,t} \leqslant 0, \tag{4}$$

$$I_{i,t} \geqslant 0, \quad x_{i,t} \geqslant 0, \quad y_{i,t} \in \{0, 1\}. \tag{5}$$

The objective function in (1) is the sum of ordering and inventory holding cost for all items over the planning horizon of length T. Since we do not allow backorders, the per unit production cost are neglected. Eq. (2) is the inventory balance equation. Requirement constraint (3) consist of the external demand when end-items are considered, and result from the lot-sizes of the immediate successors for components. We allow also external demand for components, i.e. components may be sold to outside buyers or there may be an internal demand caused by planned lots of successors outside the considered planning horizon. Constraint (4), with M being a large number, captures the fact that a setup cost is incurred whenever a batch is purchased or produced. Finally, constraint (5) states that backorders are not allowed and production being nonnegative.

3. ASMLLS algorithm

We develop the ASMLLS algorithm in several steps starting from the randomized cumulative Wagner–Whitin (RCWW) method, which was introduced by Dellaert and Jeunet (see [11]). As a first extension we use different modified setup costs for each product depending on the actual position in the production sequence (STVS—sequence-dependent time-varying setup cost). Then we add a MMAS to search for a good production sequence, and finally we introduce a systematic search for a good modification of the setup cost.

3.1. RCWW

In this section, we will discuss the concept and the influence of the time-varying modified setup cost approach. Several authors have suggested methods of modifying setup costs when solving MLLS by a series of SLLS (e.g. [21]). Dellaert and Jeunet introduced the concept of time-varying modified setup cost in their RCWW algorithm (see [11]).

For the RCWW the setup cost for each product is modified based on the fact that lot-sizing this product in the current period enforces new lots for products in previous periods. This means, if we decide to deliver product i in period t, then this fact results in an additional demand for some predecessor j in the period $t - l_i$, when the production of i starts. Now we consider two cases: (i) if product j has already a positive demand in that period, no additional costs arise; (ii) if there is no positive demand for product j we have to produce it and we must add the (modified) setup cost. So using the following notation:

$S_{i,t}$ modified setup cost of product i in period t

$T(i, j, t)$ binary variable taking the value of 1 if a lot-size of item i, delivered in period t, generates a new lot for item j in period $t - l_i$, when $j \in \Gamma^{-1}(i)$. $T(i, j, t)$ is equal to zero if there is already a positive demand for item j in period $t - l_i$ (this demand results from a planned lot-size for another successor of item j)

r $U[0, 1]$, uniformly distributed random variable

we can formulate the time-varying modified setup cost as

$$S_{i,t} = s_i + r\sum_{j\in\Gamma_i^{-1}} T(i, j, t)\frac{S_j}{|\Gamma_j|}, \tag{6}$$

where S_j can be calculated recursively by

$$S_i = s_i + \sum_{j \in \Gamma_i^{-1}} \frac{S_j}{|\Gamma_j|}. \tag{7}$$

The variable $T(i, j, t)$ is not fixed a priori, but it depends on the sequence in which the lots of the different items are determined. That means that if we perform the Wagner–Whitin (WW) algorithm (cf. [5]) for a particular item, we generate automatically new demands for its predecessors in certain periods. If we want to lot-size the next item which has a common predecessor with the previously lot-sized item, we have to take into account that not every additional demand for that predecessor results in a new lot. Hence, in Eq. (6) the setup costs are modified if and only if new lots would be necessary for some predecessor and therefore additional setup costs occur.

The factor r is a random variable which is used to weight the influence of the setup cost of the predecessors. Note that the modified setup costs depend on the sequence of the products lot-sized. Partly the sequence is determined by the bill of materials, but if there are several possibilities, the products are picked at random with a probability proportional to their unmodified setup cost s_i. Each of these products is lot-sized using a WW algorithm with the modified setup costs.

The RCWW algorithm produces several solutions according to the above description with different r-values and selects the best solution obtained.

3.2. RCWW-STVS

We extend the RCWW approach from Section 3.1 by systematically selecting different r for different products. We select two parameters $R \in [0, 0.5]$ and $u \in [-1, 1]$ e.g. randomly or adaptively. (In Section 3.3 we use a systematic search for the best pair (R, u).) For each product i the factor r_i is calculated based on the current position $\Phi_i \in \{1, \ldots, P\}$ of the product in the lot-sizing sequence:

$$r_i = R\left(1 + \frac{P - 2\Phi_i + 1}{P - 1}u\right). \tag{8}$$

Note that for the first product lot-sized ($\Phi_i = 1$, some end item) we obtain $r = R(1 + u)$ while for the last product the factor is $r = R(1 - u)$. Thus, R is the average value of r over all products and u determines the slope. The modified setup costs are now

$$S_{i,t} = s_i + r_i \sum_{j \in \Gamma_i^{-1}} T(i, j, t)\frac{S_j}{|\Gamma_j|}. \tag{9}$$

Depending on the sign of u we can distinguish between two different cases: (i) if u is positive, the factor r_i decreases with each product in the lot-sizing sequence; (ii) if u is negative, this factor increases. So in the first case we use higher modified setup costs for the first items and lower setup costs for the intermediate items. (For the raw materials we do not change the setup cost because they have no predecessors.) This leads to a lower number of lots, but larger lots for the first items. In the second case, the first products will be produced in more periods, but with smaller lot-sizes.

Example 1 (*data from Dellaert and Jeunet [11]*). We use a product structure as given in Fig. 1 with 9 products. We consider a time horizon of 9 periods, where the demand for item 1 in each period is $\langle 0, 0, 0, 0, 10, 15, 10, 12, 20\rangle$ and the demand for item 2 is $\langle 0, 0, 0, 0, 0, 50, 40, 20, 30\rangle$. The number in the circle is the product number. On the right-hand side the setup costs are written and on the left-hand side the lead time. Unit inventory holding cost of the 9 items are $\{13, 8, 4, 4, 3, 3, 2, 1, 1\}$.

At the beginning of the procedure, we choose a sequence (for the RCWW we choose a random sequence, for the ASMLLS the ants generate the sequence) and select R and u. Since we have to consider the product structure we select $\langle 2, 5, 1, 3, 4, 6, 8, 7, 9\rangle$ as the sequence and assume $R = 0.431183$, and $u = 0.956399$. So we start with item 2 and have to determine the modified setup cost. All binary variables $T(i, j, t)$ are 1 because no lots are determined yet. The modified setup cost have to be calculated from the bottom to the top of the product structure. First, we determine the

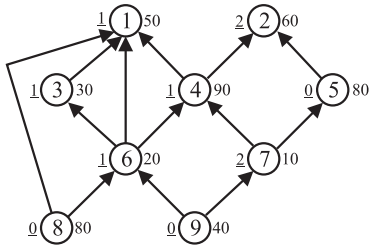

Fig. 1. Product structure used in example 1.

modified setup cost for items 8 and 9, which are always equal to the original setup costs. Then we can modify the setup costs for items 6 and 7 considering the factors r_6 and r_7, which are calculated from R and u. We continue upwards level-by-level until we can calculate the modified setup cost for item 2. In Table 1, we summarize the results for this example. The first three columns show for each product: the product number i, the original setup cost s_i and the factor r_i. Since u is positive, the r_i are decreasing from the first to the last product in the sequence. The remaining part of the table shows the modified setup costs and the lot-sizes of the products in each period. In parentheses we report the corresponding results from Dellaert and Jeunet [11]. The total cost for our solution is 2217, whereas the solution of Dellaert and Jeunet with a fixed $r = 0.431183$ (which is the same as the average r_i in our example) is 2257. Due to the changing r_i the modified setup cost for the first items (2, 5, 1, 3) are considerable larger, but for the items 4 and 6 the costs are lower. This explains also the difference in the lot-sizes. Especially for the first items we obtain fewer lots than in the solution of Dellaert and Jeunet.

Let us come back to the calculation of the $T(i, j, t)$ parameters. Assume we have already fixed the lots for items 2 and 5. The next item in the sequence is item 1. Since none of the items 3, 6, and 8 is an immediate predecessor of items 2 or 5, $T(1, 3, t)$, $T(1, 6, t)$, and $T(1, 8, t)$ are 1 for all periods. Item 4 is an immediate predecessor of item 2, and production of item 2 was scheduled for periods {6, 7, 9}. Because of the 2-periods lead time of item 2, these lots generate a demand for item 4 in the periods {4, 5, 7}. So if a new lot of item 1 generates a demand for item 4 in one of the periods {4, 5, 7}, it is not necessary to include the setup costs. Since item 1 has a lead time of one period, we set $T(1, 4, 5) = T(1, 4, 6) = T(1, 4, 8) = 0$. For the other periods we set $T(1, 4, t) = 1$.

To illustrate the influence of the choice of R and u we recalculate the example with the same lot-sizing sequence, but with $R = 0.36898$ and $u = -0.87656$. The result is shown in Table 2. The negative u leads to lower modified setup costs for the first items in the sequence and higher costs for the last items. For an example this fact causes to use 4 instead of 3 lots for item 2. The total costs are a 2257, which is equal to the total costs Dellaert and Jeunet reported, but the solution is slightly different. Fig. 2 shows the different r_i-values for the two situations calculated before.

3.3. ACO algorithm

3.3.1. Standard ACO procedure

The main idea of ACO is that a population of artificial ants repeatedly builds and improves solutions to a given instance of a combinatorial optimization problem. From one generation to the next a joint memory is updated that guides the search of the successive populations. The memory update is based on the solutions found by the ants and is biased by their associated quality.

After the initialization of the 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.

We explain the implementation of these three steps in the example of the travelling salesperson problem (TSP) which is the elementary sequencing problem. The TSP can be represented as a complete graph with a number of nodes also called cities. It is necessary to find a shortest closed tour visiting each of the cities exactly once. An artificial memory—so-called pheromone information $\tau_{ij}(m)$ is associated with each arc (i, j) connecting city i and city j. This value gives information for iteration m how desirable it was to visit customer j immediately after customer i in previous

Table 1
The modified setup costs and lot-sizing decisions in Example 1: In each line the numbers in the first half-line represent the values for the case $(R, u) = (0.431183, 0.956399)$, in which the correction factor r_i increases with i

i	s_i	r_i	Period	1	2	3	4	5	6	7	8	9
2	60	0.84	$S_{2,t}$						195.7	195.7	195.7	195.7
		(0.43)							(129.4)	(129.4)	(129.4)	(129.4)
			$X_{2,t}$						50	60		30
									(50)	(40)	(20)	(30)
5	80	0.74	$S_{5,t}$				91.1	91.1	91.1	91.1	91.1	91.1
		(0.43)					(86.5)	(86.5)	(86.5)	(86.5)	(86.5)	(86.5)
			$X_{5,t}$				50	60		30		
							(50)	(60)		(30)		
1	50	0.64	$S_{1,t}$					128.6	128.6	170.6	128.6	170.6
		(0.43)						(103.2)	(103.2)	(103.2)	(103.2)	(131.6)
			$X_{1,t}$					10	25		12	20
								(10)	(15)	(10)	(12)	(20)
3	30	0.53	$S_{3,t}$				44.3	30	30	44.3	30	30
		(0.43)					(41.5)	(30)	(30)	(30)	(30)	(30)
			$X_{3,t}$				10	25		12	20	
							(10)	(25)		(12)	(20)	
4	90	0.43	$S_{4,t}$				96.5	90	90	96.5	90	96.5
		(0.43)					(96.5)	(90)	(90)	(90)	(90)	(96.5)
			$X_{4,t}$				60	85		62		
							(60)	(55)	(30)	(62)		
6	20	0.33	$S_{6,t}$			39.7	39.7	26.6	26.6	39.7	26.6	26.6
		(0.43)				(45.9)	(45.9)	(28.6)	(28.6)	(28.6)	(28.6)	(28.6)
			$X_{6,t}$			70	120	25	74	32	20	
						(70)	(90)	(45)	(84)	(32)	(20)	
8	80	0.22	$S_{8,t}$		80	80	80	80	80	80	80	80
		(0.43)			(80)	(80)	(80)	(80)	(80)	(80)	(80)	(80)
			$X_{8,t}$		70	155		131		52		
					(70)	(145)		(141)		(52)		
7	10	0.12	$S_{7,t}$			12.4	10	10	10	10	10	10
		(0.43)				(18.6)	(10)	(10)	(10)	(10)	(10)	(10)
			$X_{7,t}$			60	135	60	62	30		
						(60)	(105)	(90)	(62)	(30)		
9	40	0.02	$S_{9,t}$	40	40	40	40	40	40	40	40	40
		(0.43)		(40)	(40)	(40)	(40)	(40)	(40)	(40)	(40)	(40)
			$X_{9,t}$	60	205	180	87	104	52			
				(60)	(175)	(180)	(107)	(114)	(52)			

The numbers in parentheses below refer to the results with constant $r = 0.431183$ as used by Dellaert and Jeunet.

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.

In the construction phase (Step 1) of the algorithm a number of ants are placed on randomly chosen cities. Then, in each construction step, each ant moves from the current city to another not yet visited city. The next city is selected on the basis of a probabilistic decision. The probabilistic choice is biased by a local heuristic information η_{ij} and the pheromone information $\tau_{ij}(m)$. In the TSP a reasonable heuristic information η_{ij} is the inverse arc length of visiting customer j after customer i. Next cities are more likely chosen when they are close to the current city and they have a high pheromone value for visiting city j immediately after city i. Each artificial ant in its current position i decides to

Table 2
Result of Example 1 in case of $(R, u) = (0.36898, -0.87656)$ in which the correction factors r_i decrease

i	s_i	r_i	Period	1	2	3	4	5	6	7	8	9
2	60	0.05	$S_{2,t}$						67.3	67.3	67.3	67.3
			$X_{2,t}$						50	40	20	30
5	80	0.13	$S_{5,t}$				81.9	81.9	81.9	81.9	81.9	81.9
			$X_{5,t}$				50	60		30		
1	50	0.21	$S_{1,t}$					75.6	75.6	75.6	75.6	89.2
			$X_{1,t}$					10	15	10	12	20
3	30	0.29	$S_{3,t}$				37.7	30	30	30	30	30
			$X_{3,t}$				10	15	10	12	20	
4	90	0.37	$S_{4,t}$				95.5	90	90	95.5	90	95.5
			$X_{4,t}$				60	55	30	62		
6	20	0.45	$S_{6,t}$			47.0	47.0	29.0	29.0	29.0	29.0	29.0
			$X_{6,t}$			70	80	55	84	32	20	
8	80	0.53	$S_{8,t}$		80	80	80	80	80	80	80	80
			$X_{8,t}$		70	145		141		52		
7	10	0.61	$S_{7,t}$			22.2	10	10	10	10	10	10
			$X_{7,t}$			60	105	90	62	30		
9	40	0.69	$S_{9,t}$	40	40	40	40	40	40	40	40	40
			$X_{9,t}$	60	175	170	117	114	52			

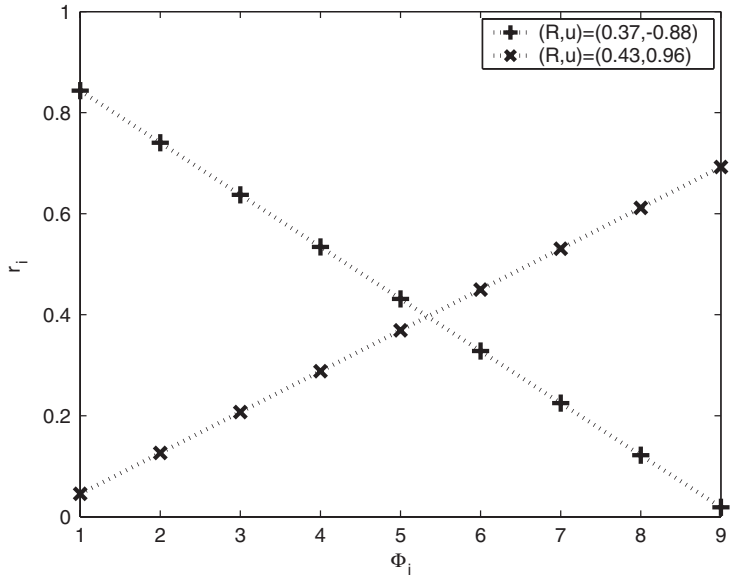

Fig. 2. Different correction factors r_i for example 1.

go to the city j with the probability

$$\mathscr{P}_{ij}^{k}(m) = \begin{cases} \dfrac{\tau_{ij}(m) \cdot \eta_{ij}}{\sum_{(l \in \mathscr{N}_i)} \tau_{il}(m) \cdot \eta_{il}} & \text{if } j \in \mathscr{N}_i^k, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Procedure ASMLLS

```
/* Initialization phase */
Generate initial R_b and u_b using RCWW-STVS
(select best solution out of 20 randomly constructed ones)
Initialize pheromone information
while (termination condition not met) do
 for each ant do
   /* Construction phase */
   Construct the production sequence according to the decision rule (12).
   /* 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 (8)
   Perform STVS (WW approximation) to evaluate the sequence
 end
 /* Pheromone update phase */
 Update the pheromone matrix according to (14), update R_b and u_b
end
```

Fig. 3. Pseudo code of ASMLLS.

The set $\mathcal{N}_i^k$ is the feasible neighborhood of the artificial ant k, which is the set of cities which have not been visited so far. In the standard ACO framework a solution obtained by the above-mentioned procedure can then be subjected to a local search in order to ensure local optimality (Step 2).

After all ants have completed their solution, the pheromone information is updated (Step 3). This is done first, by lowering the current pheromone amount by a constant factor. This procedure is also called pheromone evaporation. Furthermore, some ants with the best solution quality are allowed to update the pheromone information, i.e. depositing "artificial" pheromone on the arcs of their solution. In the MMAS only the global best ant is used to update the pheromone information after each iteration. The pheromone update rule is given by

$$\tau_{ij}(m+1) = \rho\tau_{ij}(m) + \Delta\tau_{ij}^{*}, \tag{11}$$

where $\Delta\tau_{ij}^{*} = 1/f(s^{*})$ if the global best ant (having costs $f(s^{*})$ has visited arc (i, j) and $\Delta\tau_{ij}^{*} = 0$ otherwise. To avoid too extreme differences between the pheromone values in the MMAS explicit limits $(\tau_{\min}, \tau_{\max})$ on the minimum and the maximum pheromone values are introduced; see (14) below.

3.3.2. ASMLLS

In Section 3.2, we have developed an algorithm for lot-sizing the products based on a given sequence of products which were chosen (more or less) randomly. We now add a MMAS algorithm for finding an optimal sequence of the products in order to minimize the total costs.

Each ant generates a lot-sizing sequence. As visibility η_j we use the original setup cost s_j (see Formula (12)) which is exactly what Dellaert and Jeunet [11] did in their random selection. We do not apply a local search procedure because the calculation of the total cost through the WW algorithm is very time consuming. Moreover, the results in Section 4 indicate that it is not necessary to include a local search procedure. The Pseudo code in Fig. 3 illustrates the adaptations of the standard ACO to solve the MLLS. We call this modified algorithm ASMLLS.

The algorithm starts with the initialization phase. In this phase the standard initialization routines like initializing the pheromone matrix are performed. Besides the standard initialization routines also values for R_{b} and u_{b} are generated randomly (see (8)). Within this phase the values for R_{b} and u_{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 sequences of items. Now for each sequence all the items will be lot-sized, step-by-step, by the RCWW-STVS algorithm. After all the ants have constructed their solution and the solution qualities are computed, the pheromone update is applied. As already explained in Subsection 3.3.1 in the MMAS the best solution found so far is allowed to update the artificial pheromone.

For the pheromone information we tested two different pheromone encoding schemes. We implemented the standard pheromone encoding for sequencing problems, where each element in the pheromone matrix τ_{ij} gives the desirability of lot-sizing item j immediately after item i. As alternative test design we implemented a pheromone encoding scheme which was developed for scheduling problems by Stützle [22] and also used by Merkle et al. [23]. In this pheromone encoding τ_{pj} denotes the desirability of lot-sizing item j as the pth item.

We now describe the latter approach. We consider a MLLS with P products and initial setup costs s_j. Each ant constructs a production sequence. At each step in the construction phase an ant decides 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 (12).

For the pheromone used in the decision rule we do not just consider the current pheromone value of lot-sizing item j in the current position p. Instead, we consider also all the pheromone values of lot-sizing 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. The summation rule was introduced by Merkle and Middendorf [24].

$$p_{pj}^{k}(m) = \begin{cases} \dfrac{[\sum_{o=1}^{p} \tau_{oj}(m)]^{\alpha}[s_j]^{\beta}}{\sum_{l \in N_p^k} [\sum_{o=1}^{p} \tau_{ol}(m)]^{\alpha}[s_l]^{\beta}} & \text{if } j \in N_p^k, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where $\tau_{pj}(m)$ is the intensity of pheromone trail of product j in position p at iteration m, α is the parameter to regulate the influence of $\tau_{pj}(m)$, β is the parameter to regulate the influence of s_j, N_p^k is the set of selectable products in position p of ant k based on the bill of materials.

Remark. Alternatively we also tested the standard decision rule considering only the pheromone of the current position in the sequence (see Section 4.3).

$$p_{pj}^{k}(m) = \begin{cases} \dfrac{[\tau_{pj}(m)]^{\alpha}[s_j]^{\beta}}{\sum_{l \in N_p^k} [\tau_{pl}(m)]^{\alpha}[s_l]^{\beta}} & \text{if } j \in N_p^k, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

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

$$\tau_{pj}(m+1) = \max(\tau_{\min}, \min(\tau_{\max}, \rho\tau_{pj}(m) + \Delta\tau_{pj}(m))) \tag{14}$$

with

$$\Delta\tau_{pj}(m) = \frac{1}{f(s^*)},$$

$$\tau_{\max} = \frac{1}{1-\rho} \times \frac{1}{f(s^*)},$$

$$\tau_{\min} = \frac{\tau_{\max}(1 - \sqrt[P]{0.05})}{((P/2) - 1)\sqrt[P]{0.05})},$$

where $\rho \in [0, 1]$ is the trail persistence parameter to regulate the evaporation of τ_{pj}, $\Delta\tau_{pj}(m)$ is the total increase of trail level on edge (p, j) which is controlled by maximum and minimum value along with the concept of MMAS, $f(s^*)$ is the global best solution value. The pheromone bounds are the standard MMAS values suggested in [19].

An open question is how to select R and u for applying the RCWW-STVS algorithm for lot-sizing the products and determining the costs of sequence the ant found. One could choose these parameters randomly. But tests have shown that a systematic search of optimal values for R and u gives better solutions (cf. Section 4).

This systematic search for good values for R and u is done as follows. We augmented the standard MMAS with the component of selecting R and u by using ideas of evolutionary strategies. In some initial phase, we start with randomly generating some values for R and u. Let R_b and u_b be the values that generated the best solution in this initial phase of ASMLLS. When selecting (R, u) each ant chooses randomly either (R_b, u_b) or one of the combinations $(R_b \pm \vartheta R_b, u_b \pm \vartheta u_b)$. The best solution of each iteration determines the new (R_b, u_b). In some pre-tests we have

shown that this procedure gives better solutions than choosing R and u constant by taking the best values from the initial phase.

An open question is how big ϑ should be? To find an answer for this problem we have performed some experiments and we found that ϑ should be in the interval of 0.03–0.075 and that the performance is not very sensitive with respect to this parameter. Hence we used 0.05 in our computational experiments.

4. Computational results

In this section, we turn to the description and analysis of the results obtained by our computational tests. Our ASMLLS algorithm was coded in C and compiled using Borland 6. We performed all runs on a personal computer with a Pentium 4-1.5 Ghz microprocessor, 256 MB RAM and the operating system Windows XP.

4.1. Experimental framework

We compare our algorithms with the data set used in Dellaert and Jeunet [13]. The test problems are grouped into small, medium and large-scale problem instances. The small-sized problems are composed of 4 different assembly systems (2–5 production levels), 5 items, 12 periods, 4 combinations of holding cost and setup cost and 6 demand series. This makes 96 problem instances.

The medium-sized problems include of two 50-items assembly systems (one with 9 and the other with 16 production levels) and two 40-items general systems (with communality indices[1] 1.39 and 1.18). For each problem two planning horizons are defined, 12 and 24 periods. For each combination of product structure and planning period 5 sets of setup cost and holding cost are used. This yields 40 problem instances.

The large-scale problems are composed of four 500-item general systems with 4 different communality indices (1, 1.6, 2.2, 2.8), 5 different production levels (5, 7, 9, 13, 18) and two planning horizons (12, 24) for each problem. Hence we get 40 problem instances.

4.2. MMAS parameter settings

We choose the following parameter settings for the MMAS: $\alpha=1$, $\beta=1$ and the trail persistence $\rho=0.98$ as proposed in [19,20]. We set the number of ants equal to 5. In pretest we have shown that we get almost the same results when we use a larger population (10 or 20 ants). Therefore we choose a smaller population for runtime reasons. To make results comparable, the runtimes of Dellaert and Jeunet [13] were converted according to the different machines by using the information provided by Dongarra [25].

4.3. Pretests

First, we want to analyze the contributions of our design decisions to solution quality by adding the different modifications step by step. For this purpose, we applied the different algorithms with the different modifications to all problem instances.

First we compare the results for the small instances. These instances can be solved exactly and the average objective value for all small problem instances is 810.670. We applied the following algorithms:

RCWW (average costs: 812.010). This is the original RCWW algorithm by Dellaert and Jeunet [11].
STVS (average costs: 811.197). r_i depends on a randomly chosen R and u; the lot-sizing sequence is chosen at random.
STVS-ACO* (average costs: 811.164). The lot-sizing sequence is determined by using an ACO system with standard selection rule (13).
STVS-ACO (average costs: 810.938). The *summation rule* (12) is used.
ASMLLS* (average costs: 810.924). The best R and u are searched systematically by the ants and the standard decision rule (13) is used.

[1] The communality index is the average number of successors for any item except for the end-items. ($c = \sum_{i=F+1}^{P} \Gamma(i)/(P-F)$ where F is the number of end-items.)

Table 3
Summary of total cost of various solution methods for small-sized problems

Method	Avg. costs	Best solutions found in %	Mean dev. if best solution not found (%)
LFL	1979.12	0	152.85
BPOM	972.56	19.79	26.14
SWW	879.68	4.17	10.66
CWW	887.03	32.29	14.66
PCWW	887.03	32.29	14.66
RCWW	812.01	71.88	0.78
GA	810.74	96.88	0.26
ASMLLS	810.79	92.71	0.2551
OPTIMUM	810.67	100.0	0.0

Table 4
Summary of total cost of various solution methods for medium-sized problems

Method	Avg. costs	Best solutions found in %	Mean dev. if best solution not found (%)
LFL	391,026.8	0	49.60
BPOM	305,753.2	0	16.58
SWW	292,548.7	0	10.55
CWW	280,876.7	0	6.90
PCWW	279,466.4	0	6.34
RCWW	265,561.4	0	0.76
GA	263,931.8	60	0.28
ASMLLS	263,796.3	83	0.24

ASMLLS (average costs: 810.795). The best R and u are searched systematically by the ants and the *summation rule* (12) is used.

Detailed results are reported in Table 7, 8 and 9 in the appendix. Furthermore we applied also a Wilcoxon signed-rank test with a 1% level. Table 6 shows that each extension from RCWW to the final ASMLLS is for medium (m) and large (l) test instances improves the results significantly. Only for the small problems there is no significant difference because for many of the test problems all algorithms find the optimal solution. Therefore each modification in the design has a positive influence in the solution quality. Concerning the selection rule, we have the same effect as it is commonly observed for scheduling problems: the summation rule (12) works better than the classical rule (13). Therefore we only use the summation rule for the numerical results. We should also note that we have made some preliminary tests on a subset of the smaller instances in order to find out which encoding of the pheromone is best. As expected, it turns out that an encoding of the pheromone matrix for scheduling problems *item i on position p* proposed by Stützle [19] performs better than the classical TSP encoding (i, j) i.e. *item j immediately follows item i*. Therefore, we used the specific pheromone encoding for scheduling problems in the numerical analysis of Section 4.4.

4.4. Results

We report the average solution quality, the percentage of problems in which the optimal solution (for small-sized problems) or the best known solution (for medium and large-sized problems) is found, and the percentage of deviation from the optimal (or best found) solution for the instances where the solution deviates from the best solution. Each problem instance is solved 5 times and the average results are presented. The results of the small, medium and large size problems are reported in Tables 3, 4 and 5, respectively. The optimal solutions for

Table 5
Summary of total cost of various solution methods for large-sized problems

Method	Avg. costs	Best solutions found in %	Mean dev. if best solution not found (%)
LFL	48,279,503	0	122.67
BPOM	45,743,203	0	12.11
SWW	42,383,854	0	13.58
CWW	41,179,135	0	7.71
PCWW	40,985,940	0	4.83
RCWW	41,007,946	0	3.31
GA	40,817,600	10	1.37
ASMLLS	40,371,702	90	0.179

Table 6
Statistical tests for significant difference between algorithms

	RCWW			STVS			STVS-ACO			ASMLLS*			ASMLLS		
	s	m	l	s	m	l	s	m	l	s	m	l	s	m	l
GA	<	<	<	<	<	<	<	=	=	<	=	>	=	=	>
RCWW				=	>	>	>	>	>	>	>	>	>	>	>
STVS							=	>	>	=	>	>	>	>	>
STVS-ACO										=	>	>	>	>	>
ASMLLS*													=	>	>

the small-sized problems are taken from Dellaert and Jeunet [13], obtained by GAMS using the zero-one formulation of the MLLS [26]. We compare our algorithm with the hybrid genetic algorithm developed by Dellaert and Jeunet [13] and other existing heuristics. The results for these heuristics were taken from Dellaert and Jeunet:

LFL (*lot-for-lot*)—the simple lot-for-lot rule.

BPOM (*best period order method*)—this method is based on the LFL method, but all items are ordered every θ periods, unless there are no requirements. BPOM searches for the best θ^* starting with $\theta = 1$ (which is equal to LFL) and in each step increasing θ by 1.

SWW (*sequential WW*)—application of the WW in a sequential way without any cost modification.

CWW (*cumulative WW*)—this is a sequential WW method using modified setup cost. For each item we add to the setup cost the sum of the setup costs of all its immediate predecessors.

PCWW (*partly cumulative WW*)—this method is similar to CWW, but we add only the setup costs of those predecessors for which no lot is planned so far.

RCWW (*randomized cumulative WW*)—here we add only a randomized fraction of the setup costs of all unplanned predecessors (see also Section 3.1 and Dellaert and Jeunet [11]).

For the small problem instances ASMLLS can find the optimal solution for 87 problem instances out of 96 (90.63%). We applied the algorithm for 100 iterations and the computation time is less than 1 s, which is the same runtime limit and number of iterations as for the GA. For the problems which cannot be solved to optimality we have the same deviation as the GA (0.26%).

For the medium-sized problem instances the runs consume less than 20 s CPU time (for 300 iterations) and we can find the best known solutions (which are in fact new best solutions) for 33 problem instances out of 40 (77.5%). For 7 problem instances the GA finds better solutions and for 17 cases ASMLLS and GA find the same solutions. All 7

problems, where GA is better, are assembly systems. In assembly systems the production sequence is more restricted than in general systems which reduces the decision space for the ants. Nevertheless, the deviation for these 7 problems is very small.

For the large-sized problem instances ASMLLS can find the best solutions in 36 cases out of 40 (90%). In the remaining 4 problem instances the GA outperforms the ACO algorithm. The mean deviation for these 4 cases is substantially smaller than the mean deviation of the GA for all the problem instances. The runtime is about 40 min (for 1000 iterations).

We conclude that our algorithm provides superior results in solving the medium-sized and large-sized problems in comparison to the GA solutions, while for small instances and for serial systems our approach is not better than the GA. In order to explain this, it is important to note that there is a crucial difference between the two approaches: the GA uses the Integer Program (IP) formulation as problem representation, i.e., it actually encodes the production periods for all products. By contrast we employ a two phase approach. In the first phase we construct the sequence of the products in which they should be lot-sized. For this phase of the algorithm we use ACO. In the second phase we use the STVS (WW approximation) to determine the lots for each product based on the ordering of the products. It is obvious that the decisions of the ants are very limited in serial systems, since the production sequence is predetermined through the product structure. Also for very small problems there are rather few feasible sequences and the ants cannot really contribute a lot. Consequently, the solution quality of our approach for these types of problems is not quite as good as the reference GA (which can use the whole solution space, while our approach can just produce a subset of all feasible solutions). For general product structure and larger problem sizes, however, our two phase approach outperforms the GA approach.

We verified the results through a statistical analysis where we have tested the following hypotheses using the Wilcoxon Signed Rank Test. Note that the Wilcoxon Signed Rank Test is the non-parametric counterpart of the paired samples t-test, while the Mann–Whitney U-test is the non-parametric counterpart of the standard t-test.

Hypothesis. Algorithm A outperforms Algorithm B.

$$H_0 : RPD_{\mathrm{A}} = RPD_{\mathrm{B}}$$
$$H_{\mathrm{a}} : RPD_{\mathrm{A}} < RPD_{\mathrm{B}}$$

Both hypotheses were tested using one-sided tests and our conclusion will be drawn at a 99% level of confidence. The results in Table 6 show that ASMLLS is better for large problem instances and that there is no significant difference for small- and medium-sized test instances.

5. Conclusions and outlook

A simple and efficient way to solve multi-level lot-sizing problems is cost modification. We extend a recent cost modification approach by Dellaert and Jeunet [11] by

1. optimizing the lot-sizing sequence with ACO,
2. optimizing the correction factors r_i for the cost modification for all items over the iterations.

We have shown that this combination of a metaheuristic like an ACO algorithm and a single-level lot-sizing method like Wagner–Whitin is a powerful approach for solving unconstrained multi-level lot-sizing problems. The ACO algorithm is designed to work on a large search space. If there is a very complex product structure and a high communality index, then there are many different possibilities to sequence the items for the lot-sizing step. Therefore our algorithm is better on larger and more complex problems than on smaller ones. The results for such systems are competitive. For more complex problems, ASMLLS outperforms the best methods developed so far.

Future work will focus on using a similar two step approach for capacity constrained multi-level lot-sizing problems. Furthermore, it should be noted that the application of ACO to multi-level lot-sizing problems is not restricted to our indirect two step approach. Just like in the GA by Dellaert and Jeunet [13], one could encode the complete solution in an ACO algorithm which then directly decides on the production periods for each product.

Acknowledgements

We would like to thank Nico Dellaert for providing the test data to us. Furthermore, we are grateful to Thomas Stützle for valuable comments on an earlier version of this paper. Finally, financial support from the Austrian Exchange office (ÖAD) to the first author is gratefully acknowledged.

Appendix

We now present detailed results on the contributions of the various design decisions to the solution quality. In Tables 7–9 detailed results of various solution methods for small-sized problems are given.

- Instance No.: denotes the instance number.

Table 7
Detailed results of various solution methods for small-sized problems—1/3

Instance No.	Char. (Se/Ho /De/Le)	Optimum	GA (100 it.)	ASMLLS*	ASMLLS	R	u	RCWW	STVS	STVS+ACO*	STVS+ACO
s-1	0/0/1/5	336.55	336.55	337.315	337.315	0.41	−0.52	336.55	337.315	337.315	337.315
s-2	0/0/1/4	580.6	580.6	581.03	580.6	0.19	−0.03	581.03	581.03	581.03	581.03
s-3	0/0/1/3	785.2	785.2	785.2	785.2	0.41	−0.31	785.2	785.2	785.2	785.2
s-4	0/0/1/2	984.5	984.5	984.5	984.5	0.33	0.77	984.5	984.5	984.5	984.5
s-5	0/0/2/5	342.61	342.61	342.96	342.61	0.17	0.34	343.46	345.35	343.36	343.36
s-6	0/0/2/4	592.63	592.63	592.63	592.63	0.33	−0.14	593.68	592.63	592.63	592.63
s-7	0/0/2/3	802.16	802.16	803.2	802.16	0.33	0.30	802.16	810.96	810.96	803.2
s-8	0/0/2/2	1,007.6	1,007.6	1,007.6	1,007.6	0.19	0.61	1,007.6	1,007.6	1,007.6	1,007.6
s-9	0/0/3/5	324.08	324.08	325.831	324.08	0.38	−0.82	324.69	326.831	326.831	326.831
s-10	0/0/3/4	552.76	552.76	552.76	552.76	0.22	−0.43	552.76	552.76	552.76	552.76
s-11	0/0/3/3	748.4	748.4	748.4	748.4	0.25	−0.15	748.4	748.4	748.4	748.4
s-12	0/0/3/2	928.4	928.4	928.4	928.4	0.40	0.73	928.4	928.4	928.4	928.4
s-13	0/0/4/5	344.57	344.57	344.57	344.57	0.46	−0.82	344.57	344.57	344.57	344.57
s-14	0/0/4/4	572.73	572.73	572.73	572.73	0.41	−0.41	572.73	573.883	572.73	572.73
s-15	0/0/4/3	776.16	776.16	776.16	776.16	0.28	−0.11	776.16	776.16	776.16	776.16
s-16	0/0/4/2	964.7	964.7	964.7	964.7	0.19	0.78	964.7	964.7	964.7	964.7
s-17	0/0/5/5	306.18	306.18	307.731	306.18	0.49	−0.15	307.731	307.731	307.731	307.731
s-18	0/0/5/4	523.66	523.66	523.837	523.837	0.09	−0.23	523.84	523.837	523.837	523.837
s-19	0/0/5/3	706.48	706.48	706.48	706.48	0.43	0.66	706.48	706.48	706.48	706.48
s-20	0/0/5/2	875.6	875.6	875.6	875.6	0.35	−0.36	875.6	875.6	875.6	875.6
s-21	0/0/6/5	268.06	268.06	268.06	268.06	0.10	0.45	268.18	268.06	268.06	268.06
s-22	0/0/6/4	452.05	452.05	452.05	452.05	0.29	−0.29	452.05	452.05	452.05	452.05
s-23	0/0/6/3	614.32	614.32	614.32	614.32	0.22	−0.53	614.32	614.32	614.32	614.32
s-24	0/0/6/2	764.5	764.5	764.5	764.5	0.38	0.72	764.5	764.5	764.5	764.5
s-25	0/1/1/5	501.26	501.26	502.26	502.26	0.08	0.10	514.86	502.26	502.26	502.26
s-26	0/1/1/4	759.4	759.4	759.4	759.4	0.01	−0.67	773.4	763.4	763.4	759.4
s-27	0/1/1/3	944.75	944.75	945.3	944.75	0.25	−0.65	953.75	953.75	953.75	945.3
s-28	0/1/1/2	1,115	1,115	1,115	1,115	0.44	−0.86	1,115	1,115	1,115	1,115
s-29	0/2/2/5	501.76	501.76	501.76	501.76	0.36	−0.97	508.76	501.76	501.76	501.76
s-30	0/2/2/4	765.4	765.4	765.4	765.4	0.08	0.10	772.4	765.4	765.4	765.4
s-31	0/2/2/3	956	956	956	956	0.07	−0.52	957	956	956	956
s-32	0/2/2/2	1,122	1,122	1,122	1,122	0.07	−0.52	1,122	1,122	1,122	1,122

Table 8
Detailed results of various solution methods for small-sized problems—2/3

Instance No.	Char. (Se/Ho /De/Le)	Optimum	GA (100 it.)	ASMLLS*	ASMLLS	*R*	*u*	RCWW	STVS	STVS+ACO	STVS+ACO Sum. rule
s-33	0/1/3/5	442.6	442.6	445.32	445.32	0.01	0.05	445.32	446.82	446.82	445.32
s-34	0/1/3/4	670.5	670.5	671.3	671.3	0.49	−0.98	671.3	671.3	671.3	671.3
s-35	0/1/3/3	850	850	850	850	0.32	−0.90	850	850	850	850
s-36	0/1/3/2	1,017	1,017	1,017	1,017	0.20	−0.46	1,017	1,017	1,017	1,017
s-37	0/1/4/5	452.1	452.1	452.1	452.1	0.02	0.54	454.1	452.1	452.1	452.1
s-38	0/1/4/4	687	687	687	687	0.02	0.74	687.8	687	687	687
s-39	0/1/4/3	874.75	874.75	874.75	874.75	0.07	−0.82	874.75	874.75	874.75	874.75
s-40	0/1/4/2	1,045	1,045	1,045	1,045	0.02	0.74	1,045	1,045	1,045	1,045
s-41	0/1/5/5	421.08	421.08	425.5	421.28	0.02	0.74	422.68	425.5	425.5	425.5
s-42	0/1/5/4	644.2	644.2	644.2	644.2	0.05	0.09	652.2	644.2	644.2	644.2
s-43	0/1/5/3	810.5	810.5	810.5	810.5	0.07	0.21	810.5	810.5	810.5	810.5
s-44	0/1/5/2	950	950	950	950	0.34	−0.79	950	950	950	950
s-45	0/1/6/5	377.68	377.68	377.68	377.68	0.03	0.02	398.84	377.68	377.68	377.68
s-46	0/1/6/4	571.2	571.2	571.2	571.2	0.06	−0.28	574.6	571.2	571.2	571.2
s-47	0/1/6/3	711.25	711.25	711.25	711.25	0.38	−0.94	716.25	711.25	711.25	711.25
s-48	0/1/6/2	832	832	832	832	0.39	−0.81	832	832	832	832
s-49	1/0/1/5	517.34	517.34	517.34	517.34	0.37	0.94	517.34	517.34	517.34	517.34
s-50	1/0/1/4	724.62	724.62	724.62	724.62	0.24	0.26	724.62	724.62	724.62	724.62
s-51	1/0/1/3	879.6	879.6	879.6	879.6	0.42	−0.28	879.6	879.6	879.6	879.6
s-52	1/0/1/2	1,024.5	1,024.5	1,024.5	1,024.5	0.33	0.24	1,024.5	1,024.5	1,024.5	1,024.5
s-53	1/0/2/5	525.39	525.39	525.39	525.39	0.23	0.11	525.39	525.39	525.39	525.39
s-54	1/0/2/4	739.26	739.26	739.26	739.26	0.43	0.22	739.26	739.26	739.26	739.26
s-55	1/0/2/3	898.96	898.96	898.96	898.96	0.31	0.50	898.96	898.96	898.96	898.96
s-56	1/0/2/2	1,048.7	1,048.7	1,048.7	1,048.7	0.48	0.62	1,048.7	1,048.7	1,048.7	1,048.7
s-57	1/0/3/5	507.83	507.83	507.83	507.83	0.46	−0.49	507.83	507.83	507.83	507.83
s-58	1/0/3/4	707.32	707.32	707.32	707.32	0.31	0.50	707.32	707.32	707.32	707.32
s-59	1/0/3/3	856.72	856.72	856.72	856.72	0.24	0.38	856.72	856.72	856.72	856.72
s-60	1/0/3/2	988.4	988.4	988.4	988.4	0.47	0.12	988.4	988.4	988.4	988.4
s-61	1/0/4/5	535.28	535.28	535.28	535.28	0.34	0.94	535.28	535.28	535.28	535.28
s-62	1/0/4/4	757.23	757.23	757.23	757.23	0.47	0.24	757.23	757.23	757.23	757.23
s-63	1/0/4/3	909.76	909.76	909.76	909.76	0.37	0.67	909.76	909.76	909.76	909.76
s-64	1/0/4/2	1,024.7	1,024.7	1,024.7	1,024.7	0.22	0.04	1,024.7	1,024.7	1,024.7	1,024.7

- Char. (Se/Ho/De/St): problem characteristic:
  - Se = set-up costs (0-low cost, 1-high cost).
  - Ho = holding costs (0 - low cost, 1 - high cost).
  - De = demand pattern (6 different patterns).
  - Le = number of levels in the product structure(5 levels = serial structure).

- Optimum: in this column the optimum solution is reported.
- GA: in this column the solution of the genetic algorithm is reported after 100 iterations.
- ASMLLS*: in this column the solution quality of the ACO algorithm with decision rule (13) is reported.
- ASMLLS: in this column the solution quality of the ACO algorithm with decision rule (12) is reported.
- *R*: *R*-value for the best solution found with ASMLLS.

Table 9
Detailed results of various solution methods for small-sized problems—3/3

Instance No.	Char. (Se/Ho /De/Le)	Optimum	GA (100 it.)	ASMLLS*	ASMLLS	R	u	RCWW	STVS	STVS+ACO	STVS+ACO Sum. rule
s-65	1/0/5/5	488.43	488.43	488.43	488.43	0.22	0.26	488.43	488.43	488.43	488.43
s-66	1/0/5/4	672.05	672.05	672.05	672.05	0.21	0.79	672.05	672.05	672.05	672.05
s-67	1/0/5/3	810.08	810.08	810.08	810.08	0.47	0.55	810.08	810.08	810.08	810.08
s-68	1/0/5/2	935.6	935.6	935.6	935.6	0.15	0.41	935.6	935.6	935.6	935.6
s-69	1/0/6/5	444.14	444.14	444.14	444.14	0.46	−0.08	444.14	444.14	444.14	444.14
s-70	1/0/6/4	591.52	591.52	591.52	591.52	0.26	0.21	591.52	591.52	591.52	591.52
s-71	1/0/6/3	703.6	703.6	703.6	703.6	0.29	0.53	703.6	703.6	703.6	703.6
s-72	1/0/6/2	804.5	804.5	804.5	804.5	0.38	0.77	804.5	804.5	804.5	804.5
s-73	1/1/1/5	1,172	1,172	1,174.5	1,172	0.23	−0.37	1,177.36	1,174.5	1,174.5	1,174.5
s-74	1/1/1/4	1,318	1,318	1,318	1,318	0.13	−0.47	1,318	1,318	1,318	1,318
s-75	1/1/1/3	1,269	1,269	1,269	1,269	0.12	−0.21	1,274.5	1,269	1,269	1,269
s-76	1/1/1/2	1,250	1,250	1,250	1,250	0.36	−0.73	1,250	1,250	1,250	1,250
s-77	1/1/2/5	1,153.52	1,159.52	1,159.8	1,159.8	0.12	−0.01	1,160.52	1,159.8	1,159.8	1,159.8
s-78	1/1/2/4	1,302.4	1,302.4	1,302.4	1,302.4	0.05	−0.11	1,302.4	1,302.4	1,302.4	1,302.4
s-79	1/1/2/3	1,257.5	1,257.5	1,257.5	1,257.5	0.23	−0.59	1,257.5	1,257.5	1,257.5	1,257.5
s-80	1/1/2/2	1,262	1,262	1,262	1,262	0.18	−0.71	1,262	1,262	1,262	1,262
s-81	1/1/3/5	984.92	984.92	984.92	984.92	0.03	−0.81	984.92	984.92	984.92	984.92
s-82	1/1/3/4	1,131.4	1,131.4	1,131.4	1,131.4	0.08	−0.33	1,131.4	1,131.4	1,131.4	1,131.4
s-83	1/1/3/3	1,121.25	1,121.25	1,121.25	1,121.25	0.43	−0.75	1,121.25	1,121.25	1,121.25	1,121.25
s-84	1/1/3/2	1,129	1,129	1129	1,129	0.25	−0.52	1,129	1,129	1,129	1,129
s-85	1/1/4/5	1,001.44	1,001.44	1,001.44	1,001.44	0.07	−0.44	1,005.44	1,001.44	1,001.44	1,001.44
s-86	1/1/4/4	1,146.8	1,146.8	1,146.8	1,146.8	0.24	−0.78	1,150.8	1,146.8	1,146.8	1,146.8
s-87	1/1/4/3	1,149.5	1,149.5	1,149.5	1,149.5	0.41	−0.75	1,149.5	1,149.5	1,149.5	1,149.5
s-88	1/1/4/2	1,162	1,162	1,162	1,162	0.17	−0.07	1,162	1,162	1,162	1162
s-89	1/1/5/5	932.96	932.96	932.96	932.96	0.33	−0.76	932.96	932.96	932.96	932.96
s-90	1/1/5/4	1,061.2	1,061.2	1,061.2	1,061.2	0.27	−0.94	1,063.6	1,061.2	1,061.2	1,061.2
s-91	1/1/5/3	1,049.5	1,049.5	1,049.5	1,049.5	0.02	−0.39	1,049.5	1,049.5	1,049.5	1,049.5
s-92	1/1/5/2	1,070	1,070	1,070	1,070	0.14	−0.91	1,070	1,070	1,070	1,070
s-93	1/1/6/5	826.88	826.88	826.88	826.88	0.27	−0.59	826.88	826.88	826.88	826.88
s-94	1/1/6/4	941.6	941.6	941.6	941.6	0.04	−0.12	941.6	941.6	941.6	941.6
s-95	1/1/6/3	927	927	927	927	0.25	−0.64	927	927	927	927
s-96	1/1/6/2	932	932	932	932	0.21	−0.88	932	932	932	932
		810.670	810.733	810.924	810.795			812.010	811.197	811.164	810.938

- u: u-value for the best solution found with ASMLLS.
- RCWW: in this column the results of the RCWW are reported (see Section 3.1)
- STVS: in this column the results of the RCWW and the STVS are reported (see Section 3.2)
- STVS+ACO*: in this column the results of the RCWW with STVS by using ACO are reported (see Section 3.3)
- STVS+ACO: in this column results are reported when the standard ACO is replaced by the ACO with the summation rule (see Formula 12).

In Table 10 detailed results for medium-sized problems are reported. Here we have problems with 10 different sets of parameter combinations (Pa) and 4 different product structures (St). The different structures are

1. assembly structure with 9 levels
2. assembly structure with 16 levels

Table 10
Detailed results of various solution methods for medium-sized problems

Instance No.	Char. St/Pa	GA (250 it.)	ASMLLS	Mean ASMLLS dev. if best solution not found %	Mean GA dev. if best solution not found %	*R*	*u*
m-1	1 / 1	196,323.6	196,262.0		0.0314	0.42	−0.69
m-2	2 / 1	179,761.5	179,761.5			0.48	0.64
m-3	1 / 2	166,165.3	166,268.0	0.0618		0.27	−0.79
m-4	2 / 2	155,948.0	155,960.0	0.0077		0.47	−0.24
m-5	1 / 3	201,230.9	201,229.4		0.0007	0.21	−0.15
m-6	2 / 3	183,219.1	183,219.1			0.28	0.82
m-7	1 / 4	188,010.3	188,560.0	0.2924		0.40	−0.83
m-8	2 / 4	136,669.3	136,669.3			0.37	−0.27
m-9	1 / 5	161,561.4	161,561.4			0.40	−0.21
m-10	2 / 5	187,042.1	187,042.1			0.37	−0.53
m-11	1 / 6	344,932.3	344,422.0		0.1482	0.46	−0.02
m-12	2 / 6	341,215.9	341,137.0		0.0231	0.33	−0.08
m-13	1 / 7	292,908.3	292,908.3			0.29	0.73
m-14	2 / 7	378,845.1	379,860.0	0.2679		0.40	0.13
m-15	1 / 8	355,111.4	355,520.5	0.1152		0.16	0.63
m-16	2 / 8	347,327.9	347,327.9			0.42	−0.41
m-17	1 / 9	325,606.6	325,606.6			0.23	0.56
m-18	2 / 9	412,654.9	413,859.0	0.2918		0.28	0.86
m-19	1 / 10	386,082.4	386,082.4			0.22	0.68
m-20	2 / 10	392,010.5	394,665.0	0.6772		0.28	0.31
m-21	3 / 1	148,126.1	148,069.0		0.0386	0.44	0.84
m-22	4 / 1	187,286.0	185,580.0		0.9193	0.22	0.80
m-23	3 / 2	198,067.4	198,059.0		0.0042	0.29	0.36
m-24	4 / 2	186,033.5	186,033.5			0.44	0.31
m-25	3 / 3	160,924.9	160,693.0		0.1443	0.43	−0.74
m-26	4 / 3	194,278.8	192,157.0		1.1042	0.39	−0.49
m-27	3 / 4	184,759.6	184,410.0		0.1896	0.27	0.30
m-28	4 / 4	137,533.7	137,533.7			0.39	−0.58
m-29	3 / 5	161,471.0	161,465.0		0.0037	0.24	0.68
m-30	4 / 5	166,379.8	166,379.8			0.47	−0.17
m-31	3 / 6	344,969.7	344,969.7			0.36	0.76
m-32	4 / 6	291,141.2	290,942.0		0.0685	0.43	−0.29
m-33	3 / 7	353,789.2	352,648.0		0.3236	0.45	0.33
m-34	4 / 7	341,769.5	337,913.0		1.1413	0.42	−0.29
m-35	3 / 8	357,580.9	356,739.0		0.2360	0.20	0.82
m-36	4 / 8	322,504.6	322,504.6			0.47	−0.22
m-37	3 / 9	411,707.3	411,509.0		0.0482	0.47	0.13
m-38	4 / 9	368,276.8	368,276.8			0.40	−0.55
m-39	3 / 10	401,732.2	401,732.2			0.48	0.50
m-40	4 / 10	306,314.9	306,314.9			0.14	0.68
		263,931.9	263,796.3	0.2449	0.2766		

3. general structure with a commonality index (average number of successors) of 1.39
4. general structure with a commonality index (average number of successors) of 1.18

We report also the percentage deviation of the solution quality from the comparing method if the best solution was not found.

In Table 11 detailed results for large-sized problems are reported. Here we have problems with 4 different commonality indeces (Co) and 5 different levels of the product structures (Le). We report also the percentage deviation of the solution quality from the comparing method if the best solution was not found.

Table 11
Detailed results of various solution methods for large-sized problems

Instance No.	Char. Co/Le	GA (1000 it.)	ASMLLS	Mean ASMLLS dev. if best solution not found %	Mean GA dev. if best solution not found %	*R*	*u*
l-1	1 / 5	597,560.1	596,796		0.1280	0.41	0.59
l-2	1 / 7	816,057.9	816,270	0.0260		0.46	−0.32
l-3	1 / 9	930,129.6	929,810		0.0344	0.43	0.62
l-4	1 / 13	943,802.8	942,650		0.1223	0.23	0.31
l-5	1 / 18	1,149,008.8	1,149,005		0.0003	0.23	0.10
l-6	1.6 / 5	8,232,914	8,226,990		0.0720	0.35	0.14
l-7	1.6 / 7	4,272,970	4,063,248		5.1614	0.27	0.04
l-8	1.6 / 9	2,704,332	2,713,095	0.3240		0.44	0.54
l-9	1.6 / 13	2,019,829.5	1,987,120		1.6461	0.18	0.27
l-10	1.6 / 18	1,561,600	1,560,030		0.1006	0.28	0.18
l-11	2.2 / 5	63,024,352	60,255,600		4.5950	0.35	0.49
l-12	2.2 / 7	14,363,085	14,237,100		0.8849	0.18	0.67
l-13	2.2 / 9	4,990,925.3	4,867,810		2.5292	0.20	0.31
l-14	2.2 / 13	2,910,203	2,920,056	0.3386		0.21	0.13
l-15	2.2 / 18	1,835,948	1,791,700		2.4696	0.17	−0.61
l-16	2.8 / 5	477,990,576	474,608,000		0.7127	0.35	0.01
l-17	2.8 / 7	18,759,589.1	18,750,600		0.0479	0.48	−0.62
l-18	2.8 / 9	7,602,969	7,602,730		0.0031	0.17	0.20
l-19	2.8 / 13	3,825,972	3,737,590		2.3647	0.14	0.50
l-20	2.8 / 18	2,367,030	2,358,460		0.3634	0.31	−0.64
l-21	1 / 5	1,187,241	1,187,090		0.0127	0.40	0.49
l-22	1 / 7	1,341,584	1,341,980	0.0295		0.49	0.58
l-23	1 / 9	1,409,284	1,400,480		0.6286	0.28	0.18
l-24	1 / 13	1,383,129	1,382,150		0.0708	0.27	0.11
l-25	1 / 18	1,660,995	1,660,860		0.0081	0.26	0.34
l-26	1.6 / 5	14,244,362	13,366,200		6.5700	0.26	0.28
l-27	1.6 / 7	8,173,013.5	7,671,040		6.5437	0.29	−0.49
l-28	1.6 / 9	4,363,448.6	4,326,410		0.8561	0.33	0.43
l-29	1.6 / 13	3,053,589.4	2,996,500		1.9052	0.27	−0.38
l-30	1.6 / 18	2,282,597	2,277,630		0.2181	0.25	0.46
l-31	2.2 / 5	105,233,016	103,581,000		1.5949	0.38	0.86
l-32	2.2 / 7	19,401,592	19,063,100		1.7756	0.39	0.73
l-33	2.2 / 9	7,416,366	7,361,610		0.7438	0.32	−0.34
l-34	2.2 / 13	4,350,512	4,320,570		0.6930	0.15	0.60
l-35	2.2 / 18	2,674,830	2,672,210		0.0980	0.12	0.75
l-36	2.8 / 5	779,783,213	772,761,000		0.9087	0.45	−0.38
l-37	2.8 / 7	33,684,858	33,524,300		0.4789	0.19	0.11
l-38	2.8 / 9	10,928,819	10,745,900		1.7022	0.26	0.10
l-39	2.8 / 13	5,628,807	5,627,990		0.0145	0.43	0.19
l-40	2.8 / 18	3,603,889.6	3,485,380		3.4002	0.18	0.30
		40,817,600	40,371,702	0.1795	1.3739		

References

[1] Billington PJ, McClain JO, Thomas LJ. Mathematical programming approaches to capacity-constrained MRP systems: review, formulation and problem reduction. Management Science 1983;29:1126–41.

[2] Tempelmeier H, Derstroff M. Mehrstufige Mehrprodukt-Losgrößen planung bei beschränkten Ressourcen und genereller Erzeugnisstruktur. OR Spektrum 1993;15:63–73.
[3] Tempelmeier H, Helber S. A heuristic for dynamic multi-item multi-level capacitated lotsizing for general product structures. European Journal of Operational Research 1994;75:296–311.
[4] Meyr H. Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. European Journal of Operational Research 2000;120(2):311–26.
[5] Wagner HM, Whitin TM. Dynamic version of the economic lot size model. Management Science 1958;5:89–96.
[6] Silver EA, Meal HC. A heuristic for selecting lot size requirements for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. Production and Inventory Management 1973;14:64–74.
[7] Yelle LE. Materials requirements lot sizing: a multilevel approach. International Journal of Production Research 1979;17:223–32.
[8] Veral EA, LaForge RL. The performance of a simple incremental lot-sizing rule in a multilevel inventory environment. Decision Sciences 1985;16:57–72.
[9] Blackburn JD, Millen RA. Improved heuristic performance in multi-stage lot sizing systems. Management Science 1982;28:44–56.
[10] Bookbinder JH, Koch LA. Production planning for mixed assembly/arborescent systems. Journal of Operations Management 1990;9:7–23.
[11] Dellaert NP, Jeunet J. Randomized multi-level lot sizing heuristics for general product structures. European Journal of Operational Research 2003;148:211–28.
[12] Kuik R, Salomon M. The multi-level lot-sizing problem: evaluation of a simulated annealing heuristic. European Journal of Operations Research 1990;45:25–37.
[13] Dellaert NP, Jeunet J. Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. International Journal of Production Research 2000;38:1083–99.
[14] Jeunet J, Jonard N. Single-point stochastic search algorithms for the multi-level lot-sizing problem. Computers & Operations Research 2005;32:985–1006.
[15] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Varela FJ, Bourgine P, editors. Proceedings of the first European conference on artificial life. Cambridge, USA: The MIT Press; 1992. p. 134–42.
[16] Dorigo M, Stützle T. Ant Colony Optimization. Cambridge, USA: MIT Press; 2004.
[17] Gutjahr WJ. A generalized convergence result for the graph-based ant system metaheuristic. Probability in the Engineering and Informational Sciences 2003;17:545–69.
[18] Stützle T, Dorigo M. A short convergence proof for a class of ACO algorithms. IEEE Transactions on Evolutionary Computation 2002;6(4): 358–65.
[19] Stützle T, Hoos H. The MAX-MIN ant system and local search for the traveling salesman problem. In: Proceedings of ICEC'97—1997 IEEE fourth international conference on evolutionary computation. New York: IEEE Press; 1997. p. 308–13.
[20] Stützle T, Hoos H. MAX-MIN ant system. Future Generation Computer Systems 2000;16:889–914.
[21] McLaren BJ. A study of multiple level lot sizing procedures for material requirements planning systems, PhD. dissertation, Purdue University, 1977.
[22] Stützle T. An ant approach to the flow shop problem, In: Proceedings of EUFIT'98. Aachen. 1998, p. 1560–4.
[23] Merkle D, Middendorf M, Schmeck H. Ant Colony optimization for resource constrained project scheduling. IEEE Transactions on Evolutionary Computation 2002;6:333–46.
[24] Merkle D, Middendorf M, An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In: Boers EJ-W, et al., editors. Applications of evolutionary computing: evoworkshops 2001. vol. 2037, Lecture Notes on Computer Science, Berlin: Springer; 1999. p. 287–96.
[25] Dongarra JJ. Performance of various computers using standard linear equations software. University of Tennessee Computer Science Technical Report. 2004. p. 85–9.
[26] McKnew MA, Saydam C, Coleman BJ. An efficient zero-one formulation of the multilevel lot-sizing problem. Decision Sciences 1991;22: 280–95.