Annals of Operations Research 41(1993)453–468

Statistical search methods for lotsizing problems

Marc Salomon and Roelof Kuik

Erasmus Universiteit Rotterdam, RSM, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

Luk N. Van Wassenhove

INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France

Abstract

This paper reports on our experiments with statistical search methods for solving lotsizing problems in production planning. In lotsizing problems the main objective is to generate a minimum cost production and inventory schedule, such that (i) customer demand is satisfied, and (ii) capacity restrictions imposed on production resources are not violated. We discuss our experiences in solving these, in general NP-hard, lotsizing problems with popular statistical search techniques like simulated annealing and tabu search. The paper concludes with some critical remarks on the use of statistical search methods for solving lotsizing problems.

Keywords: Lotsizing, simulated annealing, tabu search, mixed-integer programming.

1. Introduction

Literature contains a large variety of model formulations for lotsizing problems. The formulation we consider here is a mixed integer linear program (MILP) due to Billington et al. [4]:

MILP:

$$Z_{MILP} = \sum_{i=1}^{N} \left(\sum_{t=1}^{T} S_i y_{i,t} + h_i I_{i,t} + p_{i,t} x_{i,t} \right) \tag{1}$$

subject to $$I_{i,t-1} + x_{i,t-L_i} - d_{i,t} - \sum_{j<i} g_{i,j} x_{j,t} = I_{i,t} \qquad i = 1, \ldots, N;\ t = 1, \ldots, T, \tag{2}$$

$$\sum_{i \in \mathcal{C}_k} (a_{i,k} y_{i,t} + b_{i,k} x_{i,t}) \leq C_{k,t} \qquad k = 1, \ldots, K;\ t = 1, \ldots, T, \tag{3}$$

$$x_{i,t} \leq M y_{i,t} \qquad i = 1, \ldots, N;\ t = 1, \ldots, T, \tag{4}$$

$$x_{i,t}, I_{i,t} \geq 0 \qquad i = 1, \ldots, N;\ t = 1, \ldots, T, \tag{5}$$

$$y_{i,t} \in \{0,1\} \qquad i = 1, \ldots, N;\ t = 1, \ldots, T. \tag{6}$$

In this model formulation N is the number of products, T the number of time periods, and K the number of capacitated production resources. Other relevant data for the model are listed in table 1. For $i = 1, \ldots, N$ and $t = 1, \ldots, T$, decision variables are defined to represent (i) production quantities ($x_{i,t}$), (ii) end-of-period inventory positions ($I_{i,t}$), and (iii) setups ($y_{i,t}$, where $y_{i,t} = 1$ when product i is produced in period t, and $y_{i,t} = 0$ otherwise).

Table 1

List of data in MILP.

Symbol	Explanation
$a_{i,k}$	setup time of product i on production resource k (in time units)
$b_{i,k}$	production speed of product i on production resource k (in time units per production unit)
$\mathcal{C}_k$	the set of products produced by resource k
$C_{k,t}$	capacity of production resource k in period t (in time units)
$d_{i,t}$	*independent* demand for product i in period t (in production units)
$g_{i,j}$	'*gozinto factor*', representing the number of units of product i required to produce on unit of product j. Throughout this paper we assume products to be numbered such that product i can only be a component of product j when $j < i$, i.e. $g_{i,j} = 0$ when $i \geq j$
h_i	inventory holding costs for product i (in monetary units per product unit per period)
L_i	fixed production lead time (number of periods) for product i
M	some large number
$p_{i,t}$	variable production costs for product i in period t (in monetary units per production unit)
S_i	fixed (setup) cost for product i (in monetary units)

The objective (1) of MILP expresses that one is searching for a production schedule in which the sum of setup costs, inventory holding costs, and production costs is minimized. Restrictions (2) are the so-called "balance equations", which ensure that for each product–period combination (i,t) there is a balance between the inventory positions, the independent demand ($d_{i,t}$), and the dependent demand ($g_{i,j}x_{j,t}$) generated by products $j < i$. In (2) we implicitly assume for each product initial inventory at $t = 0$ and ending inventory at $t = T$ are equal to zero, i.e. $I_{i,0} = 0$ and $I_{i,T} = 0$ for $i = 1, \ldots, N$. The capacity restrictions (3) state that the total production and setup time should not exceed the available production time for production resource k in period t. The coupling between setup and production variables is modelled by

(4), while restrictions (5) are the non-negativity constraints on production and inventory variables. Finally, (6) represents the binary character of decisions on setups.

MILP is not just an interesting research problem, it is also of great practical importance. It appears, for instance, within MRP systems when trying to control production and inventory related costs, while generating *feasible* production schedules with respect to production resource requirements. However, MILP is not very tractable from a computational point of view. Arkin et al. [3] show that the optimization problem is NP-hard for general multilevel product structures[1] even in the absence of capacity restrictions and setup times. Furthermore, finding a feasible solution with respect to restrictions (2)–(6) is NP-complete when setup times are non-zero, even for the single level, single resource case [18]. Computational results show that exact solution procedures for MILP become very time consuming for larger problem instances. Therefore, most of the literature on algorithms for solving MILP focuses either on (i) exact solution procedures for restricted (polynomially solvable) problem variants, or (ii) approximation algorithms for NP-hard variants of the problem. Table 2 provides literature references on algorithms for solving some special cases of MILP, considered in this paper. For a detailed overview on algorithms for the general problem the reader is referred to Salomon [21].

Table 2

References on problems considered in this paper.

Problem description	Abbreviation	Literature
Multilevel lotsizing problem without capacity restrictions	MLLP	Afentakis et al. [1] Rosling [20] Afentakis and Gavish [2] Kuik and Salomon [15]
Multilevel capacitated lotsizing problem without setup times	MLCLP	Billington et al. [5] Billington et al. [6] Kuik et al [16]
Single level single resource capacitated lotsizing problem with setup times	SLCLPST	Trigeiro et al. [23] Cattrysse et al. [8]

This paper is further organized as follows. In section 2 we discuss the general concepts underlying the application of Simulated Annealing (SA) and Tabu Search (TS) to MILP. Specific implementations suggested by Kuik and Salomon [15], Kuik

[1]The term *multilevel* refers to the case in which end products require input from components. If this is not the case, the problem is termed "*single level*". Multilevel product structures are termed "*assembly*" if each product i is component of at most one product j, i.e. for product i $g_{ij} > 0$ for at most one product j ($j < i$). Product structures are termed "*serial*" if they are assembly and, in addition, each product j has at most one component i ($j < i$). Other product structures are usually "*general*".

et al. [16], and Cattrysse et al. [8] are discussed in section 3. Finally, we conclude with some critical remarks on the application of statistical search techniques to lotsizing.

2. Statistical search algorithms

In order to explain the general concepts underlying the application of statistical search algorithms to MILP we introduce the following notation. Let $y = (y_{1,1}, \ldots, y_{1,t} \ldots, y_{N,1}, \ldots, y_{N,t})$ be the vector of setup variables. For *fixed* y^f MILP reduces to a linear program (LP) of the form:

LP:

$$Z_{LP}(y^f) = \min \sum_{i=1}^{N} \left(\sum_{t=1}^{T} h_i I_{i,t} + p_{i,t} x_{i,t} \right) \tag{1'}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{C}_k} b_{i,k} x_{i,t} \leq C_{k,t} - \sum_{i \in \mathcal{C}_k} a_{i,k} y^f_{i,t} \qquad k = 1, \ldots, K; t = 1, \ldots, T, \tag{3'}$$

$$x_{i,t} = 0 \text{ when } y^f_{i,t} = 0 \qquad i = 1, \ldots, N; t = 1, \ldots, T, \tag{4'}$$

and (2), (5).

It should be noted that $Z_{LP}(y)$ can be computed in polynomial time for each fixed vector of setup variables y^f [14]. However, as will be shown in section 3, for many variants of MILP the corresponding LP can be solved more efficiently, by use of special purpose algorithms.

The general idea behind the statistical search algorithms is to (randomly) generate a sequence of setup patterns $y^{(0)} \rightarrow y^{(1)} \rightarrow \ldots \rightarrow \ldots \rightarrow y^{(R)}$, such that $\mathcal{F}(y) \stackrel{def}{=} \sum_{i=1}^{N} \sum_{t=1}^{T} S_i y_{i,t} + Z_{LP}(y)$ is a good approximation of Z_{MILP} in at least one of the R iterations.[2] In order to generate the sequence of setup patterns, we use simulated annealing and tabu search.

2.1. SIMULATED ANNEALING

Simulated annealing is a very simple nondeterministic optimization technique which constructs a sequence of setup patterns y (*a walk or a path*), through the set of permissible setup patterns called the *state space*. The procedure for deciding which solution to step to next is called *transition mechanism*, denoted *TM*. This *TM* requires as input the current solution, and some extra input specific to the SA

[2] Note that, by definition, $Z_{MILP} = \min_y \{\mathcal{F}(y)\}$. Furthermore, note also that in case of an infeasible setup pattern y, $Z_{LP}(y) = \infty$, which implies that $\mathcal{F}(y) = \infty$.

implementation at hand. This extra input concerns information on the set of potential solutions reachable from the current solution, called the *neighborhood* of the current solution, the determination of the *likelihood* of considering each neighbor and the setting of an *acceptance probability* for the selected neighbor. Usually, for minimization problems, after selecting a candidate neighbor (by invoking *TM*), the acceptance probability (p^{accept}) is computed as

$$p^{accept} = \begin{cases} \exp[-\beta(\mathcal{F}(y^{new}) - \mathcal{F}(y^{old}))] & \text{if } \mathcal{F}(y^{new}) > \mathcal{F}(y^{old}), \\ 1 & \text{otherwise,} \end{cases}$$

where $\mathcal{F}(y^{new})$ is the value of the objective function at the candidate neighbor solution and $\mathcal{F}(y^{old})$ is the value of the objective function at the current solution.

Here the likelihood p is drawn form a $U(0,1)$ distribution, and the walk proceeds to the candidate neighbor when $p \leq p^{accept}$. Otherwise, the neighbor is rejected and *TM* is invoked again. A subwalk during which the parameter β is held constant, is called *a chain* and the chain length is denoted by $\mathcal{L}$. The *control parameter* β is nonnegative and increases after every $\mathcal{L}$ steps of the walk. Although many variants for updating β have been proposed, we choose to update β for each chain according to,

$$\beta^{new} = \beta^{old}/\alpha \quad \text{with } 0 < \alpha < 1.$$

The newly introduced constant α is called the *descent parameter*. Finally, one has to specify a *stopping criterion*. Here, many variants have also been proposed, but we choose to specify the number of chains to be evaluated as our stopping criterion. For more (theoretical) details on SA the reader is referred to Van Laarhoven and Aarts [17], and the review by Glover and Greenberg [13].

2.2. TABU SEARCH

The tabu search method for solving MILP proceeds in a spirit analogous to SA in that it also involves a walk through the state space of feasible production schedules as defined by the feasible setup patterns. Suppose it has come to setup pattern y^{old}. Then, from the set of neighbors, a subset containing k neighbors that are not in a tabu list (to be described below) are randomly generated using transition mechanism *TM* and placed in a solution list, $\mathcal{S}_k$. The new setup pattern, y^{new} is determined as:

$$y^{new} = \{y \in \mathcal{S}_k \mid \mathcal{F}(y) \text{ is minimal on } \mathcal{S}_k\}.$$

Note that although y^{new} leads to the best solution in $\mathcal{S}_k$, it is not necessarily better than the current solution y^{old}. The role of the tabu list $\mathcal{T}_m$, which contains the m most recently visited setup patterns, is to avoid cycling in the state space (but only

to a certain extent, since a cycle, longer than the number of solutions in the tabu list, cannot be excluded). Finally, as for the SA algorithm, TS stops after a prespecified number of setup patterns have been evaluated. For further details on tabu search (applications) the reader is referred to, e.g. Glover [12] and de Werra and Hertz [9].

3. Specific implementations

3.1. THE MULTILEVEL LOTSIZING PROBLEM WITHOUT CAPACITY RESTRICTIONS (MLLP)

Simulated annealing procedures for MLLP, which is defined by (1), (2), and (4)–(6), are proposed by Kuik and Salomon [15]. Implementation specific are (i) the procedure used to solve the linear program ($\mathbf{LP_1}$) which results after fixing a setup pattern y, and (ii) the transition mechanism.

The procedure used to solve $\mathbf{LP_1}$ (which is defined by (1′), (2′), and (5)) relies on solving its dual, which is a so-called *Leontief substitution system* [24][3]. The dual of $\mathbf{LP_1}$ ($\widehat{\mathbf{LP}}_1$) is formulated as:

$\widehat{\mathbf{LP}}_1$

$$Z_{\widehat{LP}_1}(y^f) = \max \sum_{i=1}^{N} \sum_{t=1}^{T} d_{i,t}\pi_{i,t} \tag{7}$$

subject to
$$-\pi_{i,t} + \pi_{i,t+1} \le h_i \qquad i = 1,\ldots,N;\ t = 1,\ldots,T-1, \tag{8a}$$

$$-\pi_{i,T} \le h_i \qquad i = 1,\ldots,N, \tag{8b}$$

$$\pi_{i,t} - \sum_{j=1}^{N} g_{j,i} \le p_{i,t} \qquad i = 1,\ldots,N;\ t = 1,\ldots,T-1;\ y^f_{i,t} = 1. \tag{9}$$

Since $d_{i,t}$ is nonnegative an optimal solution to $\widehat{\mathbf{LP}}_1$ can be constructed by increasing the decision variable $\pi_{i,t}$ until one or more of the constraints (8a), (8b), (9) becomes binding. An optimal solution is therefore obtained by computing the values $\pi_{i,t}$ iteratively, starting with $t = 1$ and computing for $i = N, \ldots, 1$:

$$\pi_{i,1} = p'_{i,1} + \sum_{j=1}^{N} g_{j,i}\pi_{j,1}$$

and for $t = 2, \ldots, T$, $i = N, \ldots, 1$

$$\pi_{i,t} = \min\left(h_i + \pi_{i,t-1}, p'_{i,t} + \sum_{j=1}^{N} g_{j,i}\pi_{j,t}\right),$$

[3] Without loss of generality we assume lead times $L_i = 0$ for all products i.

where $p'_{i,t} = p_{i,t}$ when $y^f_{i,t} = 1$ and $p'_{i,t} = \infty$ otherwise. As the procedure above runs in $O(NT)$, $Z_{LP}(y)$ can quickly be evaluated for any fixed setup pattern y^f.

In the study summarized here experiments were conducted with *six* different transition mechanisms. They varied between *simple* ones (in which the setup pattern of the candidate neighbor differs from the current setup pattern with respect to one product-period combination only)[4] to more *elaborate* ones, in which the product structure is taken more explicitly into account.

Table 3

The table demonstrates the relation between CPU-time and quality of the SA solution for the *simple* transition mechanism (for the specific SA parameter settings used in this experiment, the reader is referred to the original paper). Assembly type problems are generated with $N = 20$ and $T = 12$ ($\mathbf{P_1}$) and $N = 20$ and $T = 20$ ($\mathbf{P_2}$). Optimal solutions to $\mathbf{P_1}$ and $\mathbf{P_2}$ have been obtained using the special purpose code by Rosling [20] for assembly type problems. For different average ratios between the optimal solution ($Z^{optimal}$) and the SA solution (Z^{SA}) the CPU-time on a SUN 3/160 workstation is given in seconds.

Problem	$\frac{Z^{SA}}{Z^{optimal}} \approx 1.09$	$\frac{Z^{SA}}{Z^{optimal}} \approx 1.07$	$\frac{Z^{SA}}{Z^{optimal}} \approx 1.03$
$\mathbf{P_1}$	158	234	353
$\mathbf{P_2}$	242	381	501

Computer programs for the simulated annealing approach are written in FORTRAN and run on a SUN3/160 workstation. Computational experiments are performed on (randomly generated) medium and larger sized problem instances, with 20–57 products, 12–20 planning periods, different setup and inventory holding cost structures, and different demand patterns for end products.

From the computational results it appears that on average (i) the more elaborate transition mechanisms do not perform better than the simple ones, (ii) for *general* product structures the quality of the solutions obtained by SA is rather good when compared to other well-known heuristics for this case (like a level-by-level application of the Wagner–Whitin [25] algorithm, combined with a cost-adaptation procedure due to Blackburn and Millen [7]), (iii) for *assembly* product structures Roslin's code outperforms *all* SA implementations with respect to computational speed and quality of the solutions[5], (iv) in order to generate "good" quality solutions the

[4] More precisely, in the simple transition mechanism the candidate neighbor setup pattern y^{new} is constructed from the current setup pattern y^{old} as follows: $y^{new}(i,t) = y^{old}(i,t)$ when $(i,t) \neq (i^*,t^*)$ and $y^{new}(i^*,t^*) = 1 - y^{old}(i^*,t^*)$, where the product number $i^* \sim DU(1,N)$ and the period number $t^* \sim DU(1,T)$ are generated at random from the discrete-uniform (DU) probability distribution.

[5] SA algorithms require on average 2–20 times the CPU-time required by Rosling's code to obtain a (near) optimal solution.

computation times required for the SA algorithms grow fast with N and T, as well as with the complexity of the product structure (table 3).

3.2. A SPECIAL CASE OF THE MULTILEVEL SINGLE RESOURCE CAPACITATED PROBLEM WITHOUT SETUP TIMES (MLCLP)

The MLCLP formulation considered by Kuik et al. [16] is a restricted version of MILP in that (i) production costs $p_{i,t}$ do not depend on t, (ii) there is only one capacitated resource ($K = 1$), (iii) setup times $a_{i,k} = 0$ for all products, and (iv) the capacity restriction occurs at a single level in the product structure (fig. 1). Restrictions (ii), (iii), and (iv) cause (3) to change to:

$$\sum_{i \in \mathcal{C}} b_i x_{i,t} \leq C_t \quad t = 1, \ldots, T, \tag{3''}$$

where $\mathcal{C}$ is the set of products at the capacitated level, b_i the capacity absorption coefficient for product i, and C_t the capacity available in period t. The other restrictions (2) and (4)–(6) of MILP remain unchanged.

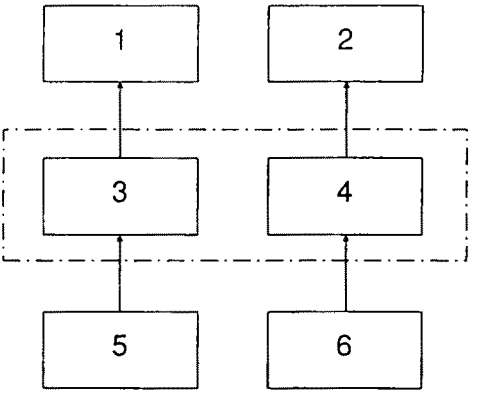

Fig. 1. A three level product structure with a capacity restriction at the second level.

As we learned from the experiments conducted for MLLP that problem specific transition mechanisms do not perform significantly better than simple ones, we have limited ourselves to simple ones for MLCLP. However, specific to the implementation of SA and TS to MLCLP is the (heuristic) solution procedure that is applied to solve

the linear program ($\mathbf{LP_2}$)[6] which results from fixing the setup pattern y^f. Problem $\mathbf{LP_2}$ was studied by McClain et al. [19]. For the special case in which (4′) is relaxed, they suggest a *greedy algorithm* which solves $\mathbf{LP_2}$ optimally. However, in $\mathbf{LP_2}$ it is necessary to compute the value of the objective function for *arbitrary* setup patterns. As far as we know, no efficient solution procedure for solving $\mathbf{LP_2}$ is available, other than an LP-based algorithm suggested by McClain et al. [19]. This algorithm, which uses Dantzig–Wolfe decomposition, is shown to require less computation time than straightforward application of the simplex method, but it is still too time consuming to be used repeatedly in a simulated annealing algorithm. For this reason Kuik et al. [16] suggest an algorithm to find an approximation $Z^A_{LP_2}(y)$ to $Z_{LP_2}(y)$.

The algorithm uses the following two observations: (i) for arbitrary setup patterns lot-for-lot production at the uncapacitated levels may no longer be possible, and (ii) there exists an optimal solution to MLCLP, in which production at the *uncapacitated levels* satisfies the *Zero-Switch Property*, which is stated as:

$$I_{i,t-1}x_{i,t} = 0 \quad \text{for} \quad i \notin \mathcal{C}.$$

A formal proof of the Zero-Switch Property is found in Kuik et al. [16].

Using the two observations stated above, the following *modified* greedy algorithm is suggested:

MODIFIED GREEDY ALGORITHM

Step 1: Start at the lowest numbered level in the product structure, not yet considered. If this level is uncapacitated, go to step 2, otherwise, go to step 3. If all levels in the product structure have been considered, then STOP.

Step 2: Plan production for each product i at the current level using the Zero-Switch Property thereby ensuring that whenever a setup is made the production quantity is large enough to cover demand in all subsequent periods in which there is no setup. Thus, if t_1 is the first period for which $y_{i,t} = 1$ and t_2 the next period for which $y_{i,t} = 1$, then set x_{i,t_1} equal to $\sum_{t=t_1}^{t_2-1}[d_{i,t} + \sum_{j<i} g_{i,j}x_{j,t}]$ and set $x_{i,t}$ equal to zero for $t = t_1 + 1, \ldots, t_2 - 1$. Proceed in this way until the end of the planning horizon.

Step 3: Plan production at the capacitated level in the following way: Select the unplanned product with the highest holding cost per unit. Plan demand for this product as late as possible, thereby putting the production quantity $x_{i,t}$ equal to zero when $y_{i,t}$ equals zero. Update available capacity and proceed until all products at this level have been planned. If the production plan becomes *infeasible* then STOP ($p^{accept} = 0$) otherwise go to step 1. (See below for a numerical example of this step.)

[6] Problem $\mathbf{LP_2}$ is identical to LP, except that (3′) is replaced by (3″).

EXAMPLE (MODIFIED GREEDY ALGORITHM, STEP 3):

	$T=1$	$T=2$	$T=3$
Product 1 $h_1=1$	$d_{11}=4$ $y_{11}=1$	$d_{12}=3$ $y_{12}=1$	$d_{13}=6$ $y_{13}=1$
Product 2 $h_2=2$	$d_{21}=6$ $y_{21}=1$	$d_{22}=5$ $y_{22}=1$	$d_{23}=4$ $y_{23}=0$
Capacities	$C_1=12$	$C_2=10$	$C_3=8$

For this example the greedy algorithm starts to schedule production for product 2, since $h_2 > h_1$. It is tried to schedule production in the period that is as close as possible to its demand period, which implies that period 1's demand is produced in period 1, and demand of period 2 is produced in period 2. However, since in period 3 no production is allowed to take place ($y_{2,3}=0$), production for period 3 must be shifted to period 2. Thus, $x_{2,1}=6$, $x_{2,2}=9$, and $x_{2,3}=0$. The remaining production capacities are now $C_1=6$, $C_2=1$, and $C_3=8$. For product 1, demand of period 1 is scheduled in period 1, but due to capacity restrictions only 1 unit of period 2's demand can be produced in period 2. Therefore the remaining production must be shifted to period 1. Finally, demand of period 3 is produced in period 3. This yields the following production schedule: $x_{1,1}=6$, $x_{1,2}=1$, and $x_{1,3}=6$. The total associated holding cost of the schedule is $2h_1+4h_2=10$.

The modified greedy algorithm does not necessarily yield an optimal solution due to the occurrence of zero-valued setup variables in some product–period combinations. The zero-valued setup variables prevent production in these periods and therefore, in conjunction with the capacity constraint already present, effectively act as capacity restrictions active at multiple levels. Thus, the value $Z_P^A(y)$ obtained by the modified greedy algorithm is only an approximation of the true optimum $Z_P(y)$. It can be obtained quickly and can therefore be used repeatedly in a simulated annealing or tabu search algorithm. This is, however, at the expense of guaranteed convergence of the SA algorithm to an optimal solution with probability one (in the long run). The performance of the SA and TS algorithms was compared to one of the few other procedures that exist for solving this problem, i.e. the facility location approach by Maes et al. [18]. In the latter approach MLCLP is reformulated as a facility location problem (FLP)[7]. The LP-relaxation of FLP is then solved, and a feasible solution to MLCLP is constructed by rounding-off fractional variables, using a number of round-off strategies.

[7]The motivation for this reformulation is that the LP-relaxation of FLP provides in general a stronger lower bound to Z_{MLCLP} and yields fewer fractional variables than the LP-relaxation of the original MLCLP. However, this approach applies to assembly product structures only.

Computational experiments are performed on (randomly generated) medium sized problem instances with two different product structures, different demand patterns, different cost structures, and different capacity utilizations. All generated problems have twelve planning periods ($T = 12$). Both product structures have three levels, of which the middle one is capacitated. The first product structure (P_1^c) consists of six products (with two end products), whereas the second product structure (P_2^c) consists of seven products, with one end product.

The SA, TS, and LP round-off (LPRO) procedures are programmed in FORTRAN and run on a SUN 3/160 workstation. Some of the computational results are summarized in table 4.

Table 4

The table demonstrates the *average* quality of the SA, TS and LPRO procedures related to the lower bound to MLCLP obtained by solving the linear programming relaxation of FLP (LP(FLP)). The numerical value of this lower bound is represented by $Z_{LP(FLP)}$. Furthermore, the CPU-times for SA, TS, and LPRO procedures are specified in seconds on a SUN 3/160 workstation. For the specific parameter settings that apply to the reported experiments, the reader is referred to the original paper.

Problem	$\frac{Z^{SA}}{Z^{LP(FLP)}}$	$\frac{Z^{TS}}{Z^{LP(FLP)}}$	$\frac{Z^{LPRO}}{Z^{LP(FLP)}}$	CPU^{SA}	CPU^{TS}	CPU^{LPRO}
P_1^c	1.12	1.14	1.17	100	108	734
P_2^c	1.23	1.24	1.44	90	101	1709

From the computational results it is concluded that (i) SA and TS outperform the LP round-off procedures both with respect to quality of the solutions, and required CPU-time, (ii) our SA implementation performs slightly better than our TS implementation[8], and (iii) a disadvantage of both SA and TS is that they do not generate lower bounds, while the LP round-off procedures do[9].

3.3. THE SINGLE LEVEL SINGLE RESOURCE CAPACITATED LOTSIZING PROBLEM WITH SETUP TIMES (SLCLPST)

In Cattrysse et al. [8] statistical search algorithms for SLCLPST are proposed. SLCLPST is equivalent to MILP except the product structure is single level and there is a single capacitated resource ($K = 1$). In the above mentioned implementation SA and TS are embedded in a *column generation heuristic* in which newly generated columns correspond to schedule proposals for individual products. The *master problem*

[8] We recognize that for other problems and/or other implementations conclusions may be different.

[9] This observation led the authors to decide to test combinations of LP with SA and TS. However, although these combinations yield relative quality information (i.e. lower bounds), the absolute quality of the solutions (upper bounds) is not better than the pure SA or TS algorithms, while computation times turn out to be significantly larger.

in this column generation heuristic is a set partitioning problem (SPP). Mathematically, SPP is formulated as follows:

SPP:

$$Z_{SPP} = \min \sum_{i=1}^{N} \sum_{\lambda=1}^{\Lambda_i} c_i^{(\lambda)} z_i^{(\lambda)} + \sum_{t=1}^{T} M o_t \tag{10}$$

subject to

$$\sum_{\lambda=1}^{\Lambda_i} z_i^{(\lambda)} = 1 \qquad i = 1, \ldots, N, \tag{11}$$

$$\sum_{i=1}^{N} \sum_{\lambda=1}^{\Lambda i} \alpha_{i,t}^{(\lambda)} z_i^{(\lambda)} - o_t \leq C_t \qquad t = 1, \ldots, T, \tag{12}$$

$$z_i^{(\lambda)} \in \{0,1\} \qquad i = 1, \ldots, N;\ \lambda = 1, \ldots, \Lambda_i, \tag{13}$$

$$o_t \geq 0 \qquad t = 1, \ldots, T, \tag{14}$$

where Λ_i is the number of columns (scheduled proposals) generated for product i. Furthermore, $c_i^{(\lambda)}$ are the total setup and holding costs associated with the λth column generated, and decision variables $z_i^{(\lambda)} = 1$ if the λth column generated is used in the solution to the master problem, while $z_i^{(\lambda)} = 0$, otherwise. Furthermore, o_t is the total overtime in period t, and M is a (sufficiently large) penalty cost, to preclude overtime production (if possible). Finally, $a_{i,t}^{(\lambda)}$ is the total production and setup time, used by product i in period t, for the λth column generated.

New columns – that will eventually be added to the master problem – result from solving for each product i a single product capacitated lotsizing problem (**SPCLP$_i$**). The latter problem is formulated as:

SPCLP$_i$:

$$Z_{SPCLP_i}(y) = \sum_{t=1}^{T} (S_{i,t}'(u) y_{i,t} + p_{i,t}'(u) x_{i,t},) \tag{15}$$

subject to

$$a_i y_{i,t} + b_i x_{i,t} \leq C_t \qquad t = 1, \ldots T, \tag{16}$$

$$x_{i,t} \leq \left(\sum_{\tau=t}^{T} d_{i,\tau} \right) y_{i,t} \qquad t = 1, \ldots, T, \tag{17}$$

$$\sum_{\tau=1}^{t} d_{i,\tau} \leq \sum_{\tau=1}^{t} x_{i,\tau} \leq U B_{i,t} \qquad t = 1, \ldots, T, \tag{18}$$

$$y_{i,t,} \in \{0,1\} \qquad t = 1, \ldots, T, \tag{19}$$

where $u = \{u_t\}$ is the set of dual multipliers corresponding to (12) in the LP-relaxation of SPP (LP(SPP)), and $S'_{i,t}(u) = S_i - a_i u_t$, while $p'_{i,t}(u) = (T - t + 1)h_i + p_{i,t} - b_i u_t$. Note that inventory variables have been eliminated from the formulations of $\mathbf{SPCLP}_i$, by substitution of $I_{i,t} = \sum_{\tau=1}^{t}(x_{i,\tau} - d_{i,\tau})$. Furthermore, based on capacity requirements for products j ($\neq i$) the formulation is tightened by deriving an upper bound ($UB_{i,t}$) on cumulative production in (18)[10].

Unfortunately, problem $\mathbf{SPCLP}_i$ is NP-hard (see Florian et al. [19]), and as far as we know no computationally effective optimal solution procedure is available. Therefore, we solve $\mathbf{SPCLP}_i$ by a *heuristic*, yielding an approximation $Z^A_{SPCLP_i}$ to Z_{SPCLP_i}. The heuristic is summarized as follows:

HEURISTIC PROCEDURE FOR SOLVING *SPCLP*$_i$

Step 1: Sequentially generate setup patterns y to $\mathbf{SPCLP}_i$, using SA *or* TS with the simple transition mechanism (see section 3).

Step 2: Solve for a fixed setup pattern y^f the resulting problem $Z_{SPCLP_i}(y^f)$. The latter is done efficiently by a special purpose code, which exploits the structural property that for a fixed setup pattern y^f the problem is equivalent to a transportation problem in which the set of resources consists of the available production capacity in each period, and the set of sinks consists of the demand in each period.

Step 3: Let $Z^A_{SPCLP_i} = \min_f Z_{SPCLP_i}(y^f)$.

The global column generation heuristic can now be described as follows:

COLUMN GENERATION HEURISTICS

Step 1: Generate for each product i a number of starting columns, using simple heuristics, and add the columns to the master problem.

Step 2: Solve LP(SPP) (we have used LINDO's LP-code developed by Schrage [22] in order to do so). Pass the dual cost multipliers u to the subproblems $\mathbf{SPCLP}_i$.

Step 3: Compute for each product i $Z^A_{SPCLP_i}$ using SA *or* TS as described above.

Step 4: Add all columns that "*price out*" – i.e. columns for which $Z^A_{SPCLPi} < w_i -$ to the master problem. Here, $w = (w_1, \ldots, w_N)$ is the set of dual multipliers corresponding to (11). If no columns price out, then go to step 5, otherwise go to step 2.

Step 5: Perform a limited branch and bound search (using the special purpose procedure by Garfinkel and Nemhauser [11]) on the columns obtained

[10] For further details on these "*valid inequalities*" the reader is referred to Cattrysse et al. [8].

at the end of the column generation heuristic, in order to search for feasible (integer) solutions to SPP among the generated columns.

Remark

The column generation *heuristic* described here differs from the traditional column generation procedure in that the subproblems $\mathbf{SPCLP}_i$ are solved by an *approximation* procedure. This might cause the column generation heuristic to stop before reaching the lower bound $Z_{LP(SPP)}$.

The column generation heuristic is implemented in FORTRAN and runs on an IBM PS/2 Model 80 machine. The results obtained by the column generation heuristic are compared with the results obtained by a dual cost heuristic due to Trigeiro et al. [23]. The latter heuristic relies upon subgradient optimization and dynamic programming in order to compute lower bounds, and an (apparently very defective) production smoothing procedure to generate feasible solutions. The test problems in our experiments are from Trigeiro et al. and have different characteristics with respect to demand pattern, cost structure, capacity (utilization), and setup time. The size of the problems varies between 6–24 products, and 15–30 periods.

Although there were initially good arguments to expect that the column generation heuristic would outperform the dual cost heuristic, after experiments the opposite conclusion seemed to be justified. It appeared that the solutions (upper bounds) were almost of the same quality as reported by Trigeiro et al.[11] However, the computational requirements for the column generation heuristic dramatically exceeded the computational requirements for the dual cost heuristic.[12] The reasons for this are twofold: (i) apparently the SA and TS heuristic are not sufficiently effective in finding good solutions to $\mathbf{SPCLP}_i$, and (ii) the enumeration scheme which is used at the end of the column generation heuristic in order to search for feasible solutions among the columns generated appears to be less effective and too time consuming when compared to the production smoothing heuristic of Trigeiro et al.

4. Conclusions

In this paper we summarize our experiences with the application of SA and TS algorithms to lotsizing problems. Computational experiments show that the success of our efforts varies. The performance of the algorithms turns out to be strongly dependent upon a large number of interrelated factors, such as problem

[11] Average gaps between lower bounds and generated solutions (upper bounds) range between 2.2% (for low capacitated problems) to 3.9% (for highly capacitated problems).

[12] CPU-times reported in Trigeiro et al. range between 10–240 seconds. CPU-times obtained by the column generation heuristic are 2–40 times higher.

structure, choice of transition mechanism, internal parameter settings (like initial acceptance probability for SA, and length of the tabu list for TS), and choice of the stopping criterion. The observation above makes it in general impossible to predict ex-ante the quality and the required computational effort of the search algorithms discussed here. Other disadvantages we encountered with respect to SA and TS are: (i) the absence of a relative quality measure, such as lower bound, and (ii) the experimental character of both methods (parameter settings and the choice of a transition mechanism require a lot of trial-and-error, since no good and generally applicable guidelines exist). Advantages of SA and TS over a number of alternative (mathematical programming based) algorithms for solving combinatorial optimization problems are: (i) the clarity of the underlying basic concepts and their ease of implementation, and (ii) the possibility of obtaining (reasonable) solutions to a number of complex combinatorial optimization problems, when standard procedures (like decomposition and/or relaxation) fail.

Summarizing, our advice to potential users is that SA and TS are certainly worthwhile to consider when trying to solve difficult combinatorial optimization problems (like some lotsizing problems), but one should not be surprised when tailor-made (mathematical programming based) procedures yield better quality solutions and require less computational effort than general purpose statistical search techniques like SA and TS.

References

[1] P. Afentakis, B. Gavish and U. Karmarkar, Computational efficient optimal solutions to the lotsizing problem in multistage assembly systems, Manag. Sci. 30(1984)222–239.

[2] P. Afentakis and B. Gavish, Optimal lot sizing for complex product structures, Oper. Res. 34(1987)237–249.

[3] E. Arkin, D. Joneja and R. Roundy, Computational complexity of uncapacitated multi-echelon production planning problems, Oper. Res. Lett. 8(1989)61–66.

[4] P.J. Billington, J.O. McClain and L.J. Thomas, Mathematical programming approaches to capacity-constrained MRP systems: review, formulation and problem reduction, Manag. Sci. 29(1983)1126–1141.

[5] P.J. Billington, J.O. McClain and L.J. Thomas, Heuristics for multilevel lotsizing with a bottleneck, Manag. Sci. 32(1986)989–1006.

[6] P.J. Billington, J. Blackburn, J. Maes, R. Millen and L.N. Van Wassenhove, Multi-product scheduling in capacitated multi-stage serial systems, Technical Report 8908/A, Econometric Institute, Erasmus University, Rotterdam, The Netherlands (1989), forthcoming in IIE Trans.

[7] J.D. Blackburn and R.A. Millen, Improvised heuristics for multistage requirements planning systems, Manag. Sci. 28(1982)44–56.

[8] D. Cattrysse, M. Salomon, R. Kuik and L.N. Van Wassenhove, Capacitated lotsizing with setup times: A comparison between Lagrangian relaxation, simulated annealing and tabu search, Unpublished manuscript, Erasmus University, Rotterdam, The Netherlands (1991).

[9] D. de Werra and A. Hertz, Tabu search techniques: a tutorial and an application to neural networks, OR Spektrum 11(1989)131–141.

[10] M. Florian, J.K. Lenstra and A.H.G. Rinnooy Kan, Deterministic production planning: algorithms and complexity, Manag. Sci. 26(1980)12–20.

[11] R.S. Garfinkel and G.L. Nemhauser, Set partitioning problem: set covering with equality constraints, Oper. Res. 17(1969)848–856.
[12] F. Glover, Tabu search, CAAI-Report 88-3, Center for Applied Artificial Intelligence, Graduate School of Business, University of Colorado, Boulder, CO (1988).
[13] F. Glover and H.J. Greenberg, New approaches for heuristic search: a bilateral linkage with artificial intelligence, Eur. Oper. Res. 39(1989)119–126.
[14] L.G. Khachian, A polynomial time algorithm for linear programming, Dokl. Akad. Nauk SSSR 244(1979)1093–1096. [English trans. in Soviet Math. Dokl. 20(1979)191–194].
[15] R. Kuik and M. Salomon, The multi-level lotsizing problem: evaluation of a stochastic optimization heuristic, Eur. J. Oper. Res. 45(1990)25–37.
[16] R. Kuik, M. Salomon, L.N. Van Wassenhove and J. Maes, Linear programming, simulated annealing and tabu search heuristics for multilevel lotsizing with a bottleneck, Technical Report 8964/A, Econometric Institute, Erasmus University, Rotterdam, The Netherlands (1989), forthcoming in IIE Trans.
[17] P.J.M. van Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications* (Reidel, Dordrecht, The Netherlands, 1987).
[18] J. Maes, J.O McClain and L.N. Van Wassenhove, Multilevel capacitated lotsizing complexity and LP-based heuristics, Eur. J. Oper. Res. 53(1991)131–148.
[19] J.O. McClain, L.J. Thomas and E.N. Weiss, Efficient solutions to a linear programming model for production scheduling with capacity constraints and no initial stock, IIE Trans. 21(1989)144–152.
[20] K. Rosling, Optimal lot-sizing for dynamic assembly systems, Technical Report 152, Linköping Institute of Technology, Sweden (1985).
[21] M. Salomon, *Deterministic Lotsizing Models for Production Planning*, Lecture Notes in Economics and Mathematical Systems, vol. 355, ed. M. Beckmann and W. Krelle (Springer, 1991).
[22] L. Schrage, *User Manual for Linear, Integer, and Quadratic Programming with LINDO* (The Scientific Press, Redwood City, 1987).
[23] W.W. Trigeiro, L.J. Thomas and J.O. McClain, Capacitated lot sizing with setup times, Manag, Sci. 35(1989)353–366.
[24] A.F. Veinott, Minimum concave-cost solution to Leontief substitution models of multifacility inventory systems, Oper. Res. 17(1969)262–291.
[25] H. Wagner and T.H. Whitin, Dynamic version of the economic lot size model, Manag. Sci. 5(1958)88–96.