Annals of Operations Research 26(1990)29–72

ANALYSIS OF RELAXATIONS FOR THE MULTI-ITEM CAPACITATED LOT-SIZING PROBLEM

W.-H. CHEN and J.-M. THIZY*
College of Business and Economics, Washington State University, Pullman, Washington 99163, USA, and
Faculty of Administration, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5

Abstract

The multi-item capacitated lot-sizing problem consists of determining the magnitude and the timing of some operations of durable results for several items in a finite number of processing periods so as to satisfy a known demand in each period. We show that the problem is strongly NP-hard. To explain why one of the most popular among exact and approximate solution methods uses a Lagrangian relaxation of the capacity constraints, we compare this approach with every alternate relaxation of the classical formulation of the problem, to show that it is the most precise in a rigorous sense. The linear relaxation of a shortest path formulation of the same problem has the same value, and one of its Lagrangian relaxations is even more accurate. It is comforting to note that well-known relaxation algorithms based on the traditional formulation can be directly used to solve the shortest path formulation efficiently, and can be further enhanced by new algorithms for the uncapacitated lot-sizing problem. An extensive computational comparison between linear programming, column generation and subgradient optimization exhibits this efficiency, with a surprisingly good performance of column generation. We pinpoint the importance of the data characteristics for an empirical classification of problem difficulty and show that most real-world problems are easier to solve than their randomly generated counterparts because of the presence of initial inventories and their large number of items.

1. Introduction

The capacitated dynamic lot-sizing problem, most easily defined in terms of production planning, consists of determining the quantity and the timing of production for several products in a finite number of periods, so as to satisfy a known demand in each period and minimize the sum of the set-up, production and inventory costs without incurring backlogs. A production capacity is imposed in each period. A set-up cost and a linear cost of production are specified, and the inventory cost is proportional to the quantity and time carried. The costs may vary for each product and each period.

*Supported by NSF Grant ECS-8518970 and NSERC Grant OGP 0042197.

The problem is known to be NP-hard even for one item [25], and the thrust of recent research has been toward the development of heuristics [44,19,20]; their major advantage is that they provide feasible solutions in relatively short time. Concurrently, various optimal solution methods have been proposed; their basic methodology is to seek a lower bound on the value of the problem as a fathoming device in an optimal enumerative search. Such a lower bound has been calculated in past research via column generation [21], subgradient optimization [53,55], cutting-planes [8] and variables redefinition [22].

Admittedly, finding an optimal solution is a computer-intensive task. It is often argued that in most cases, the approximation of the data does not justify refinement beyond fast heuristics; however, Geoffrion [33] points out the intrinsic value of an exact solution even for approximate data, because most managerial decisions entail the comparison of various alternatives evaluated in several computations: "Not only does an optimizing capability enhance the value of most individual runs, but it also provides the opportunity to make valid comparisons between the results of different runs. This is extremely important because the conclusions reached by the planning project typically rely far more heavily on comparisons between computer runs than on runs considered individually. With "quasi-optimizing" programs, such as so-called cost calculators or simulators fitted with heuristics, one never knows whether different results are due to different inputs or to the vagaries of the computer program." In discrete optimization, it is well-known that the speed of enumeration hinges critically on the quality and the ease of computation of lower bounds, and this remark suffices to justify a quest for good relaxations; moreover, seeking a lower bound presents benefits in its own right:

- By itself, a value furnished by a heuristic says nothing about how much effort should be devoted to improving the solution obtained. On the other hand, bracketing the optimal value from above (by the value of the feasible solution) and below (by the value of the relaxed solution) gives the decision-maker some benchmark for deciding whether to continue the heuristic search or not.

- Occasionally, a lower bound is more important than a feasible solution: for example, in a litigation, a contractor claims that the prespecified cost of performing a service has been dramatically increased by some changes of specifications occurring after drawing up the contract. In this case, a feasible but not optimal cost will not absolve the defending contractor, who in all rigor must provide an optimistic bound on the cost of satisfying the modified contract and then argue that the bound is still above the contracted amount. Further consideration shows that this situation is much more common than it first seems, because similar but generally less formal debates preside over the design of many production plans.

- In the course of computing the lower bound, one often finds feasible solutions and thereby brackets the optimal solution without having to resort to a separate procedure.
- Many heuristics are based on relaxations or present striking affinities with them [5,17,47].

We focus on optimal methods, although our computational experiments do not seek optimal solutions. We avoid enumeration because all the methods that we analyze calculate the same Lagrangian relaxation values, and are therefore equivalent for enumeration (at least at first brush).

In section 2, we present the classical formulation of the problem. Section 3 briefly refines the status of the problem on the computational complexity scale with a pessimistic conclusion on the ultimate power of the heuristic approach. Section 4 analyzes and compares the various Lagrangian relaxations of the classical formulation and focuses on the relaxation of the capacity constraints, for which we compare two algorithms: subgradient optimization and column generation. Section 5 presents the new formulation, some relaxations of which yield the same accuracy as the former two algorithms. Section 6 describes the implementation of these algorithms; section 7 explains the characteristics and the generation of the data sets utilized, and section 8 discusses the computational results.

2. A classical formulation of the problem

The problem can be formulated as:

$$\mathbf{P}: \; v(P) = \min \sum_{i=1}^{I} \sum_{t=1}^{T} (p_{it} x_{it} + s_{it} y_{it} + h_{it} z_{it}) \tag{1}$$

$$\text{subject to } z_{it} = z_{i,t-1} + x_{it} - d_{it} \quad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{2}$$

$$\sum_{i=1}^{I} a_i x_{it} \le c_t \quad \text{for } t = 1,\ldots,T, \tag{3}$$

$$x_{it} \le d_{itT}\, y_{it} \quad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{4}$$

$$x_{it} \ge 0,\; z_{it} \ge 0 \quad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{5}$$

$$z_{i0} = z_i^0 \quad \text{for } i = 1,\ldots,I, \tag{6}$$

$$z_{iT} = z_i^T \quad \text{for } i = 1,\ldots,I, \tag{7}$$

$$y_{it} = 0 \text{ or } 1 \quad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{8}$$

where

I is the number of products;

T is the number of production periods;

s_{it} is the production set-up cost for product i in period t;

p_{it} is the unit production cost of product i in period t;

h_{it} is the inventory cost of one unit of product i between periods t and $t + 1$;

c_t is the production capacity in period t;

a_i is the capacity consumed by the production of one unit of product i;

d_{it} is the demand for product i in period t;

$d_{it\tau} = \sum_{j=t}^{\tau} d'_{ij}$, in which d'_{ij} is the demand for product i in period j adjusted for initial and final inventories (see below);

x_{it} is the amount of product i produced in period t;

z_{it} is the inventory of product i carried from period t to period $t + 1$;

z_i^0 is the prespecified initial inventory of product i;

z_i^T is the prespecified final inventory of product i;

y_{it} is a variable that has value 1 if $x_{it} > 0$, 0 if $x_{it} = 0$.

The model assumes that the production in one period is immediately available to the customer. The constraints of the formulation can be interpreted as follows:

(1) minimize the production and inventory costs;

(2) the difference in inventory levels at the end of period t and period $t - 1$ is equal to the amount produced minus the customer's demand in period t;

(3) the consumption of the productive resource in period t cannot exceed the capacity available;

(4) and (8) a quantity can be produced only in the periods selected for production (incurring a set-up cost);

(5) no backlogging is possible;

(6) the initial inventory is specified in advance;

(7) the final inventory is specified in advance.

Most formulations of the multi-item capacitated lot-sizing problem omit constraints (6) and (7) with the understanding that the initial and final inventories are zero. There is no loss of generality in this assumption, since we can reset for each item i the given level of initial (*respectively*, ending) inventory z_i^0 (*respectively*, z_i^T)

at zero by offsetting z_i^0 from the demands of the first periods (*respectively*, adding z_i^T to the demand of the last period), thus obtaining the adjusted demands d'_{it} for $i = 1, \ldots, I$ and $t = 1, \ldots, T$. Owing to this equivalence, we can avail ourselves of all the properties of the multi-item capacitated lot-sizing problem; however, when the notation is not cumbersome, we maintain constraints (6) and (7) to remind the reader of the generality of the proof.

To simplify some proofs, we assume without loss of generality that

$$s_{it} \geq 0 \qquad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T.$$

Otherwise, cancel s_{it} by adding its value to a constant term of the objective function.

The methods proposed in this paper can also accommodate set-up times [42,55], backlogging [58] and flexible capacities (through additional resources such as overtime) [35]. Some terms to be used in later sections are described here. We denote by $v(\bullet)$ the optimal value of problem $\bullet$ and by $F(\bullet)$ its feasible region. We consider two basic formulations of the problem, P and P'. $P_\bullet$ denotes the Lagrangian relaxation of constraints $(\bullet)$; we define the Lagrangian relaxation of the bivalent requirement $y = 0$ or 1 as the linear relaxation of the problem. A subscripted literal $P_{\bullet i}$ indicates the ith component or block of $P_\bullet$.

3. Problem complexity

Bitran and Yanasse ([13], proposition 4.2) have shown that the problem is NP-hard. We prove below that it is NP-hard in a strong sense [29]; simply stated, this result asserts that unless P = NP, we cannot hope to ever refine the accuracy of heuristics without paying an exorbitant computational price for some problems. In this sense, heuristic approaches do not dominate searches for optimum solutions. The reader not interested in the technical details of the proof may proceed to the next section without loss of continuity.

THEOREM 1

The multi-item capacitated lot-sizing problem is strongly NP-hard.

Proof

We use the notation of Garey and Johnson [29]. In [13] (proposition 4.2), the proof of NP-completeness is by exhibiting a polynomial reduction r of the Partitioning Problem, an NP-complete problem, to P. Here, we display a pseudo-polynomial reduction of the Three Partition Problem, a strongly NP-complete problem, to P. The Three Partition Problem 3-P can be stated as:

Given a finite set of $3m$ elements, an integer bound B, and some integer values c_t, $t = 1, \ldots, 3m$, with $B/4 < c_t < B/2$ and $\sum_{t=1}^{3m} c_t = mB$, can we find m disjoint sets $S_1, S_2, \ldots, S_m$ such that, for $1 \leq i \leq m$, $\sum_{t \in S_i} c_t = B$?

The corresponding instance of a multi-item capacitated lot-sizing problem is given by:

m products;

$3m$ production periods;

$s_{it} = 1$ for all products i and all periods t;

$p_{it} = 0$ for all products i and all periods t;

$h_{it} = 0$ for all products i and all periods t;

$d_{it} = 0$ for all products i and all periods t, except in the last period where $d_{i,3m} = B$;

c_t for production capacity in period t;

$a_i = 1$ for all products i;

$z_i^0 = 0$ and $z_i^T = 0$ for all products i.

It is easy to see that the value of the problem cannot be lower than $3m$, because all the capacities in the $3m$ periods must be used. If 3-P has a solution, then the value of the multi-item capacitated lot-sizing problem is exactly $3m$ and any optimal solution provides a solution of 3-P as well.

To show that the reduction f of 3-P to P is pseudo-polynomial, we apply directly lemma 4.1. of [30], where:

- f can be calculated in a number of steps that is a polynomial function of the length of the input I to 3-P (which is at least $3m$),
- $Length(f(I)) \geq Length(I)$ (since all the elements of I are also present in $f(I)$),
- $Max(f(I)) \leq 3Max(I)$ (the left-hand side is $Max\{B, 3m\}$ and the right-hand side is $3Max\{B, m\}$). □

4. Relaxations of the formulation

4.1. LINEAR RELAXATION

The linear programming relaxation P_8 of the capacitated dynamic lot-sizing problem P is obtained by replacing (8) by

$$0 \leq y_{it} \leq 1 \qquad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T. \tag{9}$$

$v(P_8)$ is generally well below the optimum value $v(P)$ of the original problem. We want to give a measure of the quality of the linear relaxation, and specifically display a worst case for the error bound.

THEOREM 2

$$v(P_8) \geq \sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it}, \tag{10}$$

$$v(P) - v(P_8) \leq \sum_{i=1}^{I} \sum_{t=1}^{T} s_{it} - \sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it}, \tag{11}$$

and these bounds are attained asymptotically.

Proof

Since s_{it} is assumed to be positive, there exists an optimum value of problem P_8 such that: $y_{it} = x_{it}/d_{itT}$. Since

$$\sum_{t=1}^{T} \frac{x_{it}}{d_{itT}} \geq \frac{1}{d_{i1T}} \sum_{t=1}^{T} x_{it} = 1, \tag{12}$$

the total set-up costs of this solution are greater than $\sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it}$, which proves (10).

To prove (11), let (x, y, z) be an optimal solution of P_8 and $(x, \lceil y \rceil, z)$ a feasible solution of P obtained by fixing the fractional components of y at 1. Then, we have the following relationship:

$$\begin{aligned} v(P) - v(P_8) &\leq \sum_{i=1}^{I} \sum_{t=1}^{T} \left(s_{it} \lceil y_{it} \rceil - s_{it} \frac{x_{it}}{d_{itT}} \right) \\ &\leq \sum_{i=1}^{I} \sum_{t=1}^{T} s_{it} - \sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it} \sum_{t=1}^{T} \frac{x_{it}}{d_{itT}} \\ &\leq \sum_{i=1}^{I} \sum_{t=1}^{T} s_{it} - \sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it} \qquad \text{(due to (12))}. \end{aligned}$$

An example in which the bounds are attained asymptotically is given below:

Inventory:

$$z_{i0} = 0 \quad \text{and} \quad z_{iT} = 0, \quad \text{for} \quad i = 1, \ldots, I.$$

Demand:

$$d_{it} = \varepsilon, \quad d_{iT} = 1/\varepsilon, \qquad \text{for} \quad i = 1, \ldots, I, \quad \text{and} \quad t = 1, \ldots, T-1.$$

Capacity:

$$c_t = I\varepsilon, \qquad \text{for} \quad t = 1, \ldots, T-1, \quad \text{and} \quad c_T = I/\varepsilon.$$

Capacity absorption:

$$a_i = 1, \qquad \text{for } i = 1, \ldots, I.$$

Costs:

$$s_{it} = 1, \quad h_{it} = 0, \quad p_{it} = 0, \qquad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T.$$

The optimal solution to this problem is a lot-for-lot schedule, namely,

$$x_{it} = d_{it}, \quad y_{it} = 1, \qquad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T,$$

and an optimal solution to the linear programming relaxation of this problem is:

$$x_{it} = d_{it}, \; y_{it} = \frac{d_{it}}{d_{iT}}, \qquad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T,$$

namely,

$$y_{it} = \begin{cases} 1 & \text{if } t = T; \\ \dfrac{\varepsilon}{1/\varepsilon + (T-t)\varepsilon} & \text{otherwise,} \end{cases} \qquad \text{for } i = 1, \ldots, I.$$

As ε approaches zero, we have the following equation:

$$\lim_{\varepsilon \to 0} y_{it} = \begin{cases} 1, & \text{if } t = T; \\ 0, & \text{otherwise;} \end{cases} \qquad \text{for } i = 1, \ldots, I.$$

Therefore,

$$\lim_{\varepsilon \to 0} v(P_8) = \sum_{i=1}^{I} s_{iT} = \sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it} \quad \text{and} \quad v(P) = \sum_{i=1}^{I} \sum_{t=1}^{T} s_{it},$$

so that the right-hand sides of (10) and (11) are attained asymptotically. □

Computational experience not reported in this paper shows that the linear relaxation provides a poor lower bound in most cases.

4.2. LAGRANGIAN RELAXATION OF THE SET-UP CONSTRAINTS

A well-known result for fixed charge linear programming asserts that the Lagrangian relaxation of the fixed charge constraints – in our case (4) – is equivalent to its linear programming relaxation. We prove this briefly for ease of reference and also because theorem 9 is based on the same principle.

THEOREM 3

$$v(P_4) = v(P_8).$$

Proof

Consider the linear relaxation P_8 and the Lagrangian relaxation of constraints (4) in P_8, which we denote by $P_{4,8}$. Let $q = (q_{it})_{i=1,\ldots,I \text{ and } t=1,\ldots,T}$ be a vector of non-negative Lagrangian multipliers for constraints (4). One obtains the Lagrangian relaxation:

$$\mathbf{P_{4,8}}: \quad v(P_{4,8}) = \max_{q \geq 0} v\left(P_{4,8}(q)\right), \tag{13}$$

with:

$$\mathbf{P_{4,8}(q)}: \; v\left(P_{4,8}(q)\right) = \min \sum_{i=1}^{I} \sum_{t=1}^{T} ((p_{it} + q_{it})x_{it} + (s_{it} - q_{it}d_{itT})y_{it} + h_{it}z_{it})$$

$$\text{subject to (2), (3), (5) – (7), (9).}$$

Since the continuous variables y_{it} are restricted only by (9), they will assume an optimal value of 0 or 1 according to whether $s_{it} - q_{it}d_{itT}$ is positive or negative. This shows that P_4 possesses the integrality property [31], which in turn proves the claim. □

4.3. LAGRANGIAN RELAXATION OF THE DEMAND CONSTRAINTS

Let $w = (w_{it})_{i=1,\ldots,I \text{ and } t=1,\ldots,T}$ be a vector of Lagrangian multipliers for constraints (2). One obtains the Lagrangian relaxation:

$$\mathbf{P_2}: \quad v(P_2) = \max_{w} v\left(P_2(w)\right),$$

with:

$$\mathbf{P_2(w)}:$$

$$v\left(P_2(w)\right) = \min \sum_{i=1}^{I} \sum_{t=1}^{T} (s_{it}y_{it} + p_{it}x_{it} + h_{it}z_{it} + w_{it}(z_{it} - z_{i,t-1} - x_{it} + d_{it}))$$

$$\text{subject to (3) – (8).}$$

$v(P_2)$ is related to the value of the linear programming relaxation $v(P_8)$, as shown in the next theorem.

THEOREM 4

$$v(P_8) \leq v(P_2) \leq v(P_8) + \sum_{t=1}^{T} \max_{i=1,\ldots,I} s_{it}. \tag{14}$$

Proof

The left inequality simply states that the value of the linear relaxation of a minimization problem is no greater than the value of any Lagrangian relaxation.

For the other inequality, let $W = \{w_{it}, i = 1, \ldots, I \text{ and } t = 1, \ldots, T-1 \mid h_{it} + w_{it} \geq w_{i,t+1}\}$ be the set of values of Lagrangian multipliers such that $v(P_2(w))$ is finite. For any value $w \in W$, there exists an optimal value of $P_2(w)$ with

$$z_{i0} = z_i^0,\ z_{iT} = z_i^T,\ z_{it} = 0 \quad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T-1. \tag{15}$$

For ease of notation, let $Z(w)$ be the constant term in the objective function of $P_2(w)$:

$$Z(w) = -\sum_{i=1}^{I} w_{i1} z_i^0 + \sum_{i=1}^{I} (w_{iT} + h_{iT}) z_i^T + \sum_{i=1}^{I} \sum_{t=1}^{T} w_{it} d_{it}.$$

Therefore:

$$v(P_2(w)) = Z(w) + \min \sum_{i=1}^{I} \sum_{t=1}^{T} (s_{it} y_{it} + (p_{it} - w_{it}) x_{it})$$

$$\text{subject to (3) – (5), (8).} \tag{16}$$

Instead of P_8, we consider the linear relaxation $P_{2,8}$ of P_2 obtained by replacing constraint (8) by (9), with the well-known duality property $v(P_{2,8}) = v(P_8)$. Since s_{it} is assumed to be positive, there is also an optimum solution of $P_{2,8}(w)$ satisfying (15) and $y_{it} = x_{it}/d_{itT}$ for $i = 1, \ldots, I$ and $t = 1, \ldots, T$. Therefore, for any value $w \in W$, $v(P_{2,8}(w))$ can be written as:

$$v(P_{2,8}(w)) = Z(w) + \min \sum_{i=1}^{I} \sum_{t=1}^{T} \left(\frac{s_{it}}{d_{itT}} + p_{it} - w_{it} \right) x_{it} \tag{17}$$

$$\text{subject to (3), (5) and}$$

$$x_{it} \leq d_{itT} \quad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T. \tag{18}$$

This problem consists of T independent continuous knapsack problems with upper bounds, one for each time period t. It is well-known that each of these problems $P_{2,8,t}(w)$ has an optimal solution where at most one variable x_{it} is strictly between

its bounds. $(x_{it}, \lceil x_{it}/d_{itT} \rceil, z_{it})$ is a feasible solution of $P_2(w)$, with a value exceeding $v(P_{2,8}(w))$ by at most $\max_{i=1,\ldots,I} s_{it}$ for each index t. In particular, for w_2, the optimal Lagrangian multiplier of P_2, we can conclude:

$$v(P_2) = v\left(P_2(w_2)\right) \leq v\left(P_{2,8}(w_2)\right) + \sum_{t=1}^{T} \max_{i=1,\ldots,I} s_{it} \leq v(P_8) + \sum_{t=1}^{T} \max_{i=1,\ldots,I} s_{it},$$

where the last inequality is by Lagrangian duality. This provides the right inequality of theorem 4. □

The next corollary shows that under mild conditions, as the number of items increases, the relative difference between $v(P_2)$ and $v(P_8)$ is asymptotically zero.

COROLLARY 1

If $(\max_{i=1,\ldots,I \text{ and } t=1,\ldots,T} s_{it})/(\min_{i=1,\ldots,I \text{ and } t=1,\ldots,T} s_{it})$ is bounded by a constant S, then a relative error bound between $v(P_2)$ and $v(P_8)$ is

$$\frac{v(P_2) - v(P_8)}{v(P_8)} \leq \frac{TS}{I}.$$

Proof

Inequalities (10) and (14) yield, respectively:

$$v(P_8) \geq \sum_{i=1}^{I} \min_{t=1,\ldots,T} s_{it} \geq I \min_{\substack{i=1,\ldots,I \\ t=1,\ldots,T}} s_{it}$$

and

$$v(P_2) - v(P_8) \leq \sum_{i=1}^{T} \max_{i=1,\ldots,I} s_{it} \leq T \max_{\substack{i=1,\ldots,I \\ t=1,\ldots,T}} s_{it};$$

thus, the corollary is proved. □

4.4. LAGRANGIAN RELAXATION OF THE CAPACITY CONSTRAINT

Let $u = (u_t)_{t=1,\ldots,T}$ be a vector of non-negative Lagrangian multipliers for constraints (3). One obtains the Lagrangian relaxation:

$$\mathbf{P_3}: \qquad v(P_3) = \max_{u \geq 0} v\left(P_3(u)\right), \tag{19}$$

with

$$\mathbf{P}_3(u): \quad v(P_3(u)) = \min \sum_{t=1}^{T} \left(\sum_{i=1}^{I} (s_{it} y_{it} + p_{it} x_{it} + h_{it} z_{it}) + u_t \left(\sum_{i=1}^{I} a_i x_{it} - c_t \right) \right)$$

subject to (2), (4) – (8).

$P_3(u)$ decomposes into I uncapacitated single product lot-sizing problems $P_{3_i}(u)$, each of them solvable in $O(T \ln T)$ [24]:

$$v\left(P_3(u)\right) = \sum_{i=1}^{I} v\left(P_{3i}(u)\right) - \sum_{t=1}^{T} u_t c_t . \tag{20}$$

Bitran and Matsuo [14] prove that as the number of products I increases, the relative difference between $v(P_3)$ and $v(P)$ becomes negligible.

THEOREM 5 [14]

Let $B_i = \sum_{t=1}^{T} s_{it} - \min_{t=1,\ldots,T} s_{it}$ and let $B(T-1)$ be the sum of the $T-1$ largest B_i. Then,

$$v(P) - v(P_3) \leq B(T-1).$$

COROLLARY 2 [14]

If $(\max_{i=1,\ldots,I \text{ and } t=1,\ldots,T} s_{it})/(\min_{i=1,\ldots,I \text{ and } t=1,\ldots,T} s_{it})$ is bounded by a constant S, then a relative error bound between $v(P)$ and $v(P_3)$ is

$$\frac{v(P) - v(P_3)}{v(P_3)} \leq \frac{T(T-1)S}{I} .$$

We have seen that, for a given number of periods, as the number of items I increases, P_2 behaves asymptotically as P_8 and P_3 as P, while $v(P_8)$ can become relatively smaller than $v(P)$, as in the example of section 4.1. However, P_3 does not dominate P_2 strictly. Consider a single-item two-period problem with demands: $d_{11} = 0$, $d_{12} = 3$, capacities: $c_1 = 2$, $c_2 = 2$, capacity absorption: $a_1 = 1$, and costs as follows: $s_{1t} = 1$, $h_{1t} = 0$, $p_{1t} = 0$ for $t = 1, 2$; for this problem, $v(P_2) = 1.5$ and $v(P_3) = 1$ (neither are optimal, since $v(P) = 2$).

The next two sections explain briefly the two relaxation methods tested in this study. These explanations are necessary to prove some results in section 5, and we give more details about their implementation in section 6.

4.5. SUBGRADIENT OPTIMIZATION

The subgradient optimization method is a systematic procedure to update the value of Lagrangian multipliers; the reader is referred to the now classical exposition of the method and its implementation by Held, Wolfe and Crowder [38]. The Lagrangian multipliers are updated at iteration k by the formula:

$$u_t^{k+1} = \max\left\{0, u_t^k + v^k\left(\sum_{i=1}^{I} a_i x_{it}^k - c_t\right)\right\}, \tag{21}$$

where

$$v^k = \lambda^k \frac{v^0 - v\,(P_3(u^k))}{\sum_{t=1}^{T}\left(\sum_{i=1}^{I} a_i x_{it}^k - c_t\right)^2}, \tag{22}$$

- $(x_{it}^k, y_{it}^k, z_{it}^k)$ is an optimal solution of $P_3(u^k)$, and
- $0 < \lambda^k \leq 2$;
- v^0 is a good approximation of the solution of the problem, initially chosen as the value of a heuristic solution, then updated in the course of the algorithm (efficient methods to perform this update are described in section 6.1).

4.6. COLUMN GENERATION PROCEDURE

Column generation, another standard procedure to update the value of the Lagrangian multipliers, was first applied to the multi-item capacitated lot-sizing problem by Dzielinski and Gomory [21]. For a description of the particular variant used here, the reader is referred to the solution of a closely related problem by Lasdon and Terjung [45]. In summary, the Lagrangian multipliers are found at each iteration as the optimal dual variables corresponding to the constraints (24) of the following linear program, which can be loosely interpreted as attempting to smooth the infeasible solutions proposed from the K previous iterations (since each product can be handled separately, we could index the indices k and K below by each item i, which we avoid doing for the sake of simplicity).

$$\mathbf{P_3^K}: \quad v\,(P_3^K) = \min \sum_{i=1}^{I}\sum_{k=1}^{K} \gamma_{ik}\,\theta_{ik} \tag{23}$$

$$\text{subject to} \sum_{i=1}^{I}\sum_{k=1}^{K} \alpha_{itk}\,\theta_{ik} \leq c_t \quad \text{for } t = 1,\ldots,T, \tag{24}$$

$$\sum_{k=1}^{K} \theta_{ik} = 1 \quad \text{for } i = 1, \ldots, I, \tag{25}$$

$$\theta_{ik} \geq 0 \qquad \text{for } i = 1, \ldots, I \text{ and } k = 1, \ldots, K, \tag{26}$$

where:

- K is the number of previous iterations,
- $\gamma_{ik} = \sum_{t=1}^{T} (s_{it} y_{it}^k + p_{it} x_{it}^k + h_{it} z_{it}^k)$ is the cost of producing the ith item in the kth uncapacitated solution,
- $\alpha_{itk} = a_i x_{it}^k$ is the amount of capacity c_t required by the kth uncapacitated solution to produce the ith item,
- the variable θ_{ik} determines the weight of the kth uncapacitated solution for the ith item.

The constraints of the formulation can be interpreted as follows:

(23) minimize the total costs;

(24) let the smoothed solution respect the capacity constraint in each period;

(25), (26) for each product i, the smoothed solution is a weighted average of the K preceding ones.

Let $(-u^K, \pi^K)$ be a set of optimal dual variables corresponding to the constraints (24) and (25) of P_3^K (u^K is non-negative). The solution of P_3^K is optimal for P_3 if the reduced cost of every column of P_3 is non-negative:

$$\gamma_{ik} + \sum_{t=1}^{T} \alpha_{itk} u_t^K - \pi_i^K = \sum_{t=1}^{T} \left(s_{it} y_{it}^k + \left(p_{it} + u_t^K a_i \right) x_{it}^k + h_{it} z_{it}^k \right) - \pi_i^K \geq 0$$

$$\text{for } i = 1, \ldots, I,$$

where k, y_{it}^k, x_{it}^k and z_{it}^k correspond to the extreme points of the polyhedron defined by the constraints (2), (4)–(8).

The above expression, which is closely related to the objective function of $P_3(u^K)$, can be evaluated by solving I independent uncapacitated single-item lot-sizing problems; if it is negative, it provides a new solution from which to generate a new column $(\gamma_{i,K+1}, \alpha_{it,K+1})$ to form P_3^{K+1}. The process reiterates as described above.

Lasdon and Terjung [45] report that the following strategy yields computational savings: first solve the subproblems by using a heuristic until no improvement is possible and then resort to an exact algorithm. Bahl [6], and Bahl and Ritzman [7] solve every uncapacitated single-item lot-sizing subproblem approximately with a

heuristic, which allows them to save time for each subproblem and also to generate fewer columns because fewer alternate uncapacitated solutions are produced by the heuristic; their computational results indicate that their algorithm is fast with only a slight loss in solution quality.

Our contribution is to show that even optimal Lagrangian values can be obtained efficiently; we expect that in a branch-and-bound algorithm, this additional increase of the lower bound will suffice to fathom many nodes.

5. Shortest path formulation

Eppen and Martin [22] give a new linear programming characterization of the problem. Pochet and Wolsey [49] propose other formulations, which they show to be less efficient than Eppen and Martin's. The latter is based on a well-known graphic representation of the problem that assigns a node per beginning of period (including period $T + 1$), and an arc between each pair of nodes $(t, \tau + 1)$ representing the possible decision to produce in period t the quantity necessary to satisfy the demands of periods t through τ. Assigning a cost of production to this arc, a shortest path formulation leads to the optimal solution.

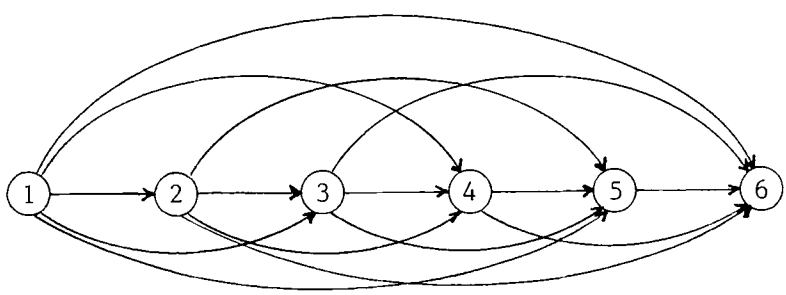

Fig. 1. A shortest path representation of the single-item lot-sizing problem.

For I products, there will be I such graphs. What links them are T capacity constraints, which can be viewed as capacities on the flow of production issued from each node. For product i, let $x_{it\tau}$ be the decision variable corresponding to an arc between nodes t and $\tau + 1$, which is also the fraction of the demands for periods t through τ of product i that is produced in period t. One can immediately calculate the amount of production from:

$$x_{it} = \sum_{\tau=t}^{T} x_{it\tau} d_{it\tau} \qquad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T, \tag{27}$$

and the inventories z_{it} by:

$$z_{it} = z_i^0 + \sum_{\tau=1}^{t} x_{i\tau} - d_{i1t} \qquad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{28}$$

and by (6). However, we cannot obtain $(x_{it\tau}, y_{it})$ from (x_{it}, y_{it}, z_{it}) so simply. The capacity required for production in period t can therefore be written as:

$$\sum_{i=1}^{I} a_i x_{it} = \sum_{i=1}^{I} a_i \left(\sum_{\tau=t}^{T} x_{it\tau} d_{it\tau} \right) = \sum_{i=1}^{I} \sum_{\tau=t}^{T} a_i d_{it\tau} x_{it\tau}. \tag{29}$$

This quantity must be no greater than the capacity c_t, so the capacity constraints (3) can be written as (32) below, in which we define $a_{it\tau} = a_i d_{it\tau}$. The set-up constraints (4) can be written as:

$$\sum_{\tau=t}^{T} x_{it\tau} d_{it\tau} \le d_{itT} y_{it} \quad \text{or} \quad \sum_{\tau=t}^{T} \frac{d_{it\tau}}{d_{itT}} x_{it\tau} \le y_{it}, \tag{30}$$

which can be strengthened to (35) below.

The shortest path formulation is:

P′:

$$v(P') = \min \sum_{i=1}^{I} \sum_{t=1}^{T} \sum_{\tau=t}^{T} f_{it\tau} x_{it\tau} + \sum_{i=1}^{I} \sum_{t=1}^{T} s_{it} y_{it} \tag{31}$$

$$\text{subject to} \sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau} \le c_t \qquad \text{for } t = 1,\ldots,T, \tag{32}$$

$$\sum_{\tau=1}^{T} x_{i1\tau} = 1 \qquad \text{for } i = 1,\ldots,I, \tag{33}$$

$$-\sum_{\tau=1}^{t-1} x_{i\tau,t-1} + \sum_{\tau=t}^{T} x_{it\tau} = 0 \quad \text{for } i = 1,\ldots,I \text{ and } t = 2,\ldots,T, \tag{34}$$

$$\sum_{\substack{\tau=t \\ \tau: a_{it\tau}>0}}^{T} x_{it\tau} \le y_{it} \qquad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{35}$$

$$x_{it\tau} \ge 0 \qquad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T \text{ and } \tau = 1,\ldots,T, \tag{36}$$

$$y_{it} = 0 \text{ or } 1 \qquad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T, \tag{37}$$

where:

$f_{it\tau}$ = total cost of producing product i in period t to satisfy demands for periods t through τ, excluding set-up costs.

In this formulation,

(31) is to minimize the total cost;

(32) are the capacity constraints over the planning horizon;

(33) and (34) are the shortest path equations;

(35) are fixed-charge constraints.

The quality of this alternate formulation hinges on the fact that its linear relaxation P'_{37} is as good as the Lagrangian relaxation P_3 above.

THEOREM 6 [48]

$$v(P) = v(P')$$

and

$$v(P'_{37}) = v(P'_{32}) = v(P_3).$$

We give a constructive proof in the sequel. The original intent of Eppen and Martin [22] is to provide a broad-based formulation to be solved with a standard linear programming code. In this setting, cutting planes are embedded to tighten the linear relaxation P'_{37}. However, since this formulation is based on a fixed-charge, multi-commodity capacitated network flow model, it is interesting to explore the potential of network flow optimization methods to solve at least the linear relaxation of this large-scale program which possesses $I(T + 1)$ nodes and $IT(T + 1)/2$ arcs. (The classical formulation gives also rise to a different fixed-charge network flow problem with $T(I + 1) + 1$ nodes and also $IT(T + 1)/2$ arcs [23] – we use it in section 6.1; the number of arcs can be reduced to $I(2T - 1) + T$.)

We are not aware of the existence of general-purpose, efficient codes to solve large instances of fixed-charge, multi-commodity capacitated network flow problems (for the special case of facility location, see [1,2,32,36,43]). On the other hand, the linear relaxation of the problems has been thoroughly investigated and various efficient methods have been proposed. Bertsekas and Gafni [10], and Bertsekas and Tseng [9] have shown conclusively that for very large networks, relaxation techniques are extremely effective. In our particular case featuring mutual capacities on multiple commodities, even for medium-sized networks, it appears that the benefits of a streamlined simplex algorithm based on partitioning and network flow data structures are not so marked as in pure networks [40,41,3,4]. Relaxation of the capacity constraints is an attractive alternative, especially when the number of mutual capacity constraints

is relatively small [41,3,4,50] (there are only T capacity constraints in our network). This can be formulated as:

$$\mathbf{P}'_{32}: \qquad v(P'_{32}) = \max_{u' \geq 0} v\left(P'_{32}(u')\right), \tag{38}$$

where

$$\mathbf{P}'_{32}(\mathbf{u}'): \quad v\left(P'_{32}(u')\right) = \min \sum_{t=1}^{T} \left(-u'_t c_t + \sum_{i=1}^{I} \sum_{\tau=t}^{T} (f_{it\tau} + u'_t a_{it\tau}) x_{it\tau} + \sum_{i=1}^{I} s_{it} y_{it} \right)$$

$$\text{subject to (33)} - \text{(37)}. \tag{39}$$

We show below that efficient solution methods are already available to us, because both subgradient optimization and column generation to solve P_3 can also be viewed as special-purpose algorithms for solving P'_{32}. We do not claim that these are the best techniques for solving P or even P'_{32}, but only that these well-known methods are good contenders to solve the new formulation.

5.1. LINEAR RELAXATIONS

Theorem 7 proves constructively that $v(P'_{32}) = v(P_3)$.

THEOREM 7

There exists an implementation of a subgradient optimization algorithm to solve P'_{32} such that every iteration is identical to that of a subgradient optimization algorithm to solve P_3.

Proof

We give below a sketch of the implementation of a subgradient optimization algorithm to solve P'_{32}: the Lagrangian multipliers are updated at iteration k by the formula:

$$u_t'^{k+1} = \max \left\{ 0, u_t'^{k} + v'^{k} \left(\sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau}^{k} - c_t \right) \right\}, \tag{40}$$

where

$$v'^{k} = \lambda'^{k} \frac{v'^{0} - v\left(P'_{32}(u'^{k})\right)}{\sum_{t=1}^{T} \left(\sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau}^{k} - c_t \right)^2}, \tag{41}$$

- $(x^k_{it\tau}, y^k_{it})$ is an optimal solution of $P'_{32}(u'^k)$,
- $0 < \lambda'^k \leq 2$, and
- v'^0 is a good approximation of the solution of P'.

To obtain identical iterations, choose the same initial values $v^0 = v'^0$, $u^0 = u'^0$ and $\lambda^0 = \lambda'^0$ for both subgradient algorithms. Given a non-negative set of Lagrangian multipliers, $P'_{32}(u'^k)$ and $P_3(u^k)$ represent the same uncapacitated lot-sizing problem and can be solved identically to provide some optimal solutions $(x^k_{it}, y^k_{it}, z^k_{it})$ and $(x^k_{it\tau}, y^k_{it})$, respectively. Recall that the update of u^k is performed via (21) and (22). In both formulae, we can apply the following identity

$$\sum_{i=1}^{I} a_i x^k_{it} - c_t = \sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x^k_{it\tau} - c_t$$

as a direct consequence of (29). The same updating scheme can be applied to λ'^k as to λ^k (see next section); therefore, (22) and (41) yield the same value $v^k = v'^k$; hence (21) and (40) provide the same updated $u^{k+1} = u'^{k+1}$ and a simple recurrence shows that all the iterations coincide. □

Theorem 8 proves constructively that $v(P'_{37}) = v(P_3)$.

THEOREM 8

There exists a simplicial decomposition algorithm to solve P'_{37} such that every iteration is identical to that of a column generation algorithm to solve P_3.

Proof

First, we note by Lagrangian duality for linear programming that $P'_{37} = P'_{32,37}$ and we will analyze the latter relaxation (in fact, we could use P'_{32} or $P'_{32,37}$ interchangeably in this as well as the preceding theorem). Applying the minimax theorem over convex sets ([52], theorems 6.7.8 and 6.2.9),

$$\mathrm{P}'_{32,37}\colon\ v(P'_{32,37}) = \min_{x,y \mid (33)-(36),(9)} \sup_{u' \geq 0} \sum_{t=1}^{T} \left(-u'_t c_t + \sum_{i=1}^{I} \sum_{\tau=t}^{T} (f_{it\tau} + u'_t a_{it\tau}) x_{it\tau} + \sum_{i=1}^{I} s_{it} y_{it} \right) \tag{42}$$

is a convex piecewise linear function of x and y on the domain $\{x_{it\tau} \mid \sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau} \leq c_t\}$. Recently, simplicial decomposition has been proposed as an effective method to solve large-scale nonlinear network flow problems [56,37]. It performs successively a *major iteration* consisting of a linear problem over the feasible region (where the

objective is a tangential approximation of the function at the incumbent feasible point), then a *minor iteration* minimizing the original function over the convex hull of the former K incumbent points. As in the preceding theorem, the reader recognizes the solution of P'_{32} (or equivalently P_3) as the major iteration and the solution of the master problem as the minor iteration. However, two technical difficulties are evoked but not addressed explicitly in [56]:

- the gradient of the piecewise linear objective is not defined at any of its angular points; following Frank and Wolfe [27], and Holloway [39], one can show that a subgradient suffices, which is precisely what the subproblem of column generation furnishes;
- at the border of the domain, $\sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau} \leq c_t$, a subgradient $\sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau} - c_t$ can be used, which again is the option followed by the master program.

Several implementations can be applied while keeping each step of the algorithms for P'_{37} and P_3 identical. We suggest two:

- drop some points from the simplex: this leads to simplicial decomposition for P'_{37} and column generation for P_3;
- drop all but one point from the simplex, which yields Frank and Wolfe's algorithm for P'_{37} and our strategy to solve P_3 (see next section). □

In addition to its good computational performance, column generation possesses an inherent advantage over subgradient optimization to implement the shortest path formulation; for a given set of variables θ_{ik}, it is easy to calculate $x_{it\tau}$;

$$x_{it\tau} = \sum_{k=1}^{K} \theta_{ik} y^{k}_{it\tau},$$

where

$$y^{k}_{it\tau} = \begin{cases} 1, & \text{if } \alpha_{itk} > 0,\ \alpha_{i,\tau+1,k} > 0 \text{ and } \alpha_{ijk} = 0 \text{ for } t < j < \tau + 1; \\ 0, & \text{otherwise.} \end{cases}$$

(Here, we make the convenient abuse of notation: $\alpha_{i,T+1,k} > 0$ for $i = 1, \ldots, I$ and $k = 1, \ldots, K$.) In contrast, as mentioned previously, $x_{it\tau}$ cannot be calculated immediately for x_{it}. This is particularly advantageous if we want to use the primal solution for a heuristic or cutting-plane generation.

The Lagrangian relaxation of the set-up constraints of P' is also equivalent to its linear relaxation.

THEOREM 9

$$v(P'_{35}) = v(P'_{37}).$$

Proof

See theorem 3. □

5.2. LAGRANGIAN RELAXATION OF THE SHORTEST PATH CONSTRAINTS

Let $w' = (w'_{it})_{i=1,\ldots,I \text{ and } t=1,\ldots,T}$ be a vector of Lagrangian multipliers for constraints (33) and (34). One obtains the Lagrangian relaxation:

$$\mathbf{P}'_{33,34}: \; v(P'_{33,34}) = \max_{w'} v(P'_{33,34}(w')),$$

where

$\mathbf{P}'_{33,34}(\mathbf{w}')$:

$$v(P'_{33,34}(w')) = \min \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{\tau=t}^{T} f_{it\tau}x_{it\tau} + \sum_{i=1}^{I}\sum_{t=1}^{T} s_{it}y_{it} + \sum_{i=1}^{I} w'_{i1}\left(\sum_{\tau=1}^{T} x_{i1\tau} - 1\right)$$
$$+ \sum_{i=1}^{I}\sum_{t=2}^{T} w'_{it}\left(-\sum_{\tau=1}^{t-1} x_{i\tau,t-1} + \sum_{\tau=t}^{T} x_{it\tau}\right)$$

subject to (32), (35) – (37).

$v(P'_{33,34})$ is related to the linear programming relaxation $v(P'_{37})$, as shown in the next theorem.

THEOREM 10

$$v(P'_{37}) \le v(P'_{33,34}) \le v(P'_{37}) + \sum_{t=1}^{T} \max_{i=1,\ldots,I} s_{it}.$$

Proof

The left inequality simply states that the value of the linear relaxation of a minimization problem is no greater than the value of any Lagrangian relaxation.

For the other inequality, instead of P'_{37}, we consider the linear relaxation $P'_{33,34,37}$ of $P'_{33,34}$ with the well-known duality property $v(P'_{33,34,37}) = v(P'_{37})$. There is an optimum solution of $P'_{33,34,37}(w')$ with

$$y_{it} = \sum_{\substack{\tau=t \\ \tau:\, a_{it\tau}>0}}^{T} x_{it\tau} \qquad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T. \tag{43}$$

Therefore, for any given set of multipliers w', $v(P'_{33,34,37}(w'))$ can be written as:

$$\mathbf{P}'_{33,34,37}(\mathbf{w}'): \quad v(P'_{33,34,37}(w')) = -\sum_{i=1}^{I} w'_{i1} + \min \sum_{i=1}^{I} \sum_{t=1}^{T} \sum_{\tau=t}^{T} f'_{it\tau} x_{it\tau}$$

subject to (32), (36) and

$$\sum_{\substack{\tau=t \\ \tau: a_{it\tau}>0}}^{T} x_{it\tau} \le 1 \quad \text{for } i = 1,\ldots,I \text{ and } t = 1,\ldots,T,$$

where

$$f'_{it\tau} = \begin{cases} f_{it\tau} + w'_{it} - w'_{i,\tau+1} + s_{it}, & \text{if } a_{it\tau} > 0; \\ f_{it\tau} + w'_{it} - w'_{i,\tau+1}, & \text{otherwise.} \end{cases}$$

(Of course, this definition holds only for $t \le \tau$; we again make a convenient abuse of notation: $w'_{i,T+1} = 0$ for $i = 1,\ldots,I$.)

This problem is formed by T independent subproblems, one for each time period t. Each independent problem $P'_{33,34,37,t}(w')$ can be written as:

$$\mathbf{P}'_{33,34,37,t}(\mathbf{w}'): \quad \text{minimize} \sum_{i=1}^{I} \sum_{\tau=t}^{T} f'_{it\tau} x_{it\tau}$$

$$\text{subject to} \sum_{i=1}^{I} \sum_{\tau=t}^{T} a_{it\tau} x_{it\tau} \le c_t,$$

$$\sum_{\substack{\tau=t \\ \tau: a_{it\tau}>0}}^{T} x_{it\tau} \le 1 \qquad \text{for } i = 1,\ldots,I,$$

$$x_{it\tau} \ge 0 \qquad \text{for } i = 1,\ldots,I \text{ and } \tau = 1,\ldots,T.$$

Since the variables $x_{it\tau}$ with $a_{it\tau} = 0$ are not bounded by any constraint, we let W' be the set of values of Lagrangian multipliers such that $v(P'_{33,34,37,t}(w'))$ is finite. Let $\tau_i = \arg\min_{\tau=t,\ldots,T} f'_{it\tau}/a_{it\tau}$. Given any $w' \in W'$:

$$\mathbf{P}'_{33,34,37,t}(\mathbf{w}'): \quad \text{minimize} \sum_{i=1}^{I} f'_{it\tau_i} x_{it\tau_i}$$

$$\text{subject to} \sum_{i=1}^{I} a_{it\tau_i} x_{it\tau_i} \le c_t,$$

$$0 \le x_{it\tau_i} \le 1 \qquad \text{for } i = 1,\ldots,I.$$

As in theorem 4, it is well known that $P'_{33,34,37,t}(w')$ has one optimal solution such that for at most one index i, a variable $x_{it\tau_i}$ is strictly between its bounds 0 and 1. Completed with 0 for $\tau \neq \tau_i$, $(x_{it\tau}, \lceil x_{it\tau} \rceil)$ is a feasible solution of $P'_{33,34}(w')$, with a value exceeding $v(P'_{33,34,37}(w'))$ by at most $\max_{i=1,\ldots,I} s_{it}$ for each index t. In particular for $w'_{33,34} \in W'$, the optimal Lagrangian multiplier of $P'_{33,34}$, we can conclude:

$$v(P'_{33,34}) = v(P'_{33,34}(w'_{33,34})) \leq v(P'_{33,34,37}(w'_{33,34})) + \sum_{t=1}^{T} \max_{i=1,\ldots,I} s_{it}$$

$$\leq v(P'_{37}) + \sum_{t=1}^{T} \max_{i=1,\ldots,I} s_{it},$$

where the last inequality is by Lagrangian duality. This provides the right inequality of theorem 10. □

COROLLARY 3

If $(\max_{i=1,\ldots,I \text{ and } t=1,\ldots,T} s_{it})/(\min_{i=1,\ldots,I \text{ and } t=1,\ldots,T} s_{it})$ is bounded by a constant S, then a relative error bound between $v(P'_{33,34})$ and $v(P'_{37})$ is

$$\frac{v(P'_{33,34}) - v(P'_{37})}{v(P'_{37})} \leq \frac{TS}{I}.$$

Proof

See corollary 1. □

Although $v(P'_{33,34})$ is the tightest relaxation studied in this paper, the previous corollary indicates that its value will be close to that of the linear relaxation as the number of items increases. Since our experimental results confirm that asymptotic results are verified in practice, we have not tried to implement a Lagrangian relaxation of the shortest path constraints.

5.3. SUMMARY OF RELAXATIONS

We summarize the results of this section and complete the analysis of the relaxations.

THEOREM 11

The following inequalities hold:

$$v(P_8) = v(P_4) \leq \begin{matrix} v(P_3) = v(P'_{32}) = v(P'_{35}) = v(P'_{37}) \\ v(P_2) \end{matrix} \leq v(P'_{33,34}) \leq v(P) = v(P'). \tag{44}$$

Every inequality is strict for at least one instance of multi-item capacitated lot-sizing problem and no other inequality exists among the values of (44) except those derived by transitivity.

Proof

All inequalites have been proved except $v(P_2) \leq v(P'_{33,34})$. To prove this, we adapt theorem 1(d) of [31]. Consider the following program:

$$\mathbf{Q}: \quad v(Q) = \text{minimize} \quad cx \tag{45}$$

$$\text{subject to} \quad Ax \leq b, \tag{46}$$

$$x \in X. \tag{47}$$

The value of the Lagrangian relaxation of Q with respect to constraints (46) is equal to the value of the following program:

$$\mathbf{Q}_{46}: \quad v(Q_{46}) = \text{minimize} \quad cx$$

$$\text{subject to} \quad Ax \leq b,$$

$$x \in [X],$$

where $[X]$ denotes the convex hull of the set bounded X. In our case, we need to show that if an element $(x_{it\tau}, y_{it})$ is in $F(P'_{33,34})$, we can always construct a feasible solution (x_{it}, y_{it}) in $F(P_2)$ with the same objective function. The construct is given by (27) (y_{it} remains the same). We need to show that if a point is in the convex hull of the feasible region defined by (32) and (35)–(37), its transform is in the convex hull of the feasible region defined by (3)–(8). In addition, if it satisfies (33) and (34), its transform satisfies (2). We first prove the following

LEMMA 1

Let $x_{it\tau}$ satisfy constraints (33) and (34) for $i = 1, \ldots, I$ and $t = 1, \ldots, T$ and $\tau = 1, \ldots, T$; then

$$\sum_{\tau=t}^{T} \sum_{k=1}^{t} x_{ik\tau} = 1 \quad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T. \tag{48}$$

Proof

This can be seen by choosing a cut for the graph of fig. 1 corresponding to a vertical line just to the left of node $t+1$. We also prove this result algebraically. For $t = 1$, (48) is simply (33). Suppose (48) is true for $1 \le t < T$. Then, $1 = \sum_{k=1}^{t} \sum_{\tau=t}^{T} x_{ik\tau} = \sum_{k=1}^{t} \sum_{\tau=t+1}^{T} x_{ik\tau} + \sum_{k=1}^{t} x_{ikt}$. The last term is $\sum_{\tau=t+1}^{T} x_{i,t+1,\tau}$ by (34). Therefore, $1 = \sum_{\tau=t+1}^{T} \sum_{k=1}^{t} x_{ik\tau} + \sum_{\tau=t+1}^{T} x_{i,t+1,\tau} = \sum_{\tau=t+1}^{T} \sum_{k=1}^{t+1} x_{ik\tau}$, which is (48) for $t+1$. □

Now we can prove the following

LEMMA 2

Let $x_{it\tau}$ satisfy constraints (33) and (34) and define x_{it} for $i = 1, \ldots, I$ and $t = 1, \ldots, T$ by (27). Then,

$$\sum_{\tau=1}^{t} x_{i\tau} = d_{i1t} + \sum_{\tau=t+1}^{T} d_{i,t+1,\tau} \sum_{k=1}^{t} x_{ik\tau} \quad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T. \tag{49}$$

Proof

For $t = 1$, (49) is satisfied because by definition (27):

$$x_{i1} = \sum_{\tau=1}^{T} x_{i1\tau} d_{i1\tau} = d'_{i1} \sum_{\tau=1}^{T} x_{i1\tau} + \sum_{\tau=2}^{T} d_{i2\tau} x_{i1\tau},$$

but $\sum_{\tau=1}^{T} x_{i1\tau} = 1$ by (33); hence,

$$x_{i1} = d'_{i1} + \sum_{\tau=2}^{T} x_{i1\tau} d_{i2\tau},$$

which is (49).

Now suppose that (49) is satisfied for $1 \le t < T$. Then,

$$\begin{aligned} \sum_{\tau=1}^{t+1} x_{i\tau} &= \sum_{\tau=1}^{t} x_{i\tau} + x_{i,t+1} \\ &= d_{i1t} + \sum_{\tau=t+1}^{T} d_{i,t+1,\tau} \sum_{k=1}^{t} x_{ik\tau} + \sum_{\tau=t+1}^{T} x_{i,t+1,\tau} d_{i,t+1,\tau} \\ &= d_{i1t} + \sum_{\tau=t+1}^{T} d_{i,t+1,\tau} \sum_{k=1}^{t+1} x_{ik\tau} \end{aligned}$$

$$= d_{i1t} + d'_{i,t+1} \sum_{\tau=t+1}^{T} \sum_{k=1}^{t+1} x_{ik\tau} + \sum_{\tau=t+2}^{T} d_{i,t+2,\tau} \sum_{k=1}^{t+1} x_{ik\tau}$$

$$= d_{i1t} + d'_{i,t+1} + \sum_{\tau=t+2}^{T} d_{i,t+2,\tau} \sum_{k=1}^{t+1} x_{ik\tau} \tag{50}$$

and (50) is (49) for $t + 1$ (the last inequality results from lemma 1). □

We are now ready to prove

LEMMA 3

$$v(P_2) \leq v(P'_{33,34}).$$

Proof

A feasible point $(x_{it\tau}, y_{it})$ of $(P'_{33,34})$ is in the convex hull of the region defined by (32) and (35)–(37), which means that there exists a family of points $(x^k_{it\tau}, y^k_{it})$ for all $k = 1, \ldots, K$, satisfying (32) and (35)–(37), such that

$$(x_{it\tau}, y_{it}) = \sum_{k=1}^{K} \lambda_k \left(x^k_{it\tau}, y^k_{it}\right),$$

$$\sum_{k=1}^{K} \lambda_k = 1, \quad \lambda_k \geq 0 \quad \text{for} \quad k = 1, \ldots, K.$$

Define x^k_{it} for $i = 1, \ldots, I$ and $t = 1, \ldots, T$ and $k = 1, \ldots, K$, by (27) (using a super-script k). By (29), (30), (x^k_{it}, y^k_{it}) satisfies constraints (3), (4), (8) and $x^k_{it} \geq 0$ for $i = 1, \ldots, I$ and $t = 1, \ldots, T$ and $k = 1, \ldots, K$. Let $x_{it} = \sum_{k=1}^{K} \lambda_k x^k_{it}$; since x_{it} and $x_{it\tau}$ are defined by the same convex combination of some extreme points, x_{it} is related to $x_{it\tau}$ by (27). By defining some inventory variables z_{it} by (6) and (28), we ensure that equations (2) and (7) are satisfied. Since $x_{it\tau}$ satisfies constraints (33) and (34), lemma 2 indicates that $z_{it} \geq 0$ for $i = 1, \ldots, I$ and $t = 1, \ldots, T$. So (x_{it}, y_{it}, z_{it}) is in the convex hull of the feasible region defined by (3)–(8), and it satisfies (2). Finally, the definition of the costs $f'_{it\tau_i}$ for $i = 1, \ldots, I$ and $t = 1, \ldots, T$ and $\tau = 1, \ldots, T$ ensures that $(x_{it\tau}, y_{it})$ and (x_{it}, y_{it}, z_{it}) have the same objective value. □

For the problem instance given at the end of section 4.4, $v(P_3) < v(P_2) = v(P'_{33,34}) < v(P)$. For the following single-item two-period problem with demands: $d_{11} = 1$, $d_{12} = 1$, capacities: $c_1 = 2$, $c_2 = 2$, capacity absorption: $a_1 = 1$ and costs as follows: $s_{1t} = 1$, $h_{1t} = 0$ for $t = 1, 2$, $p_{11} = 1$, $p_{12} = 0$, $v(P_2) = 2.5 < v(P_3) = 3$, proving the theorem. □

6. Algorithmic implementation

6.1. SUBGRADIENT OPTIMIZATION

It is generally recognized that subgradient optimization is a good approach for obtaining fast approximations of a Lagrangian relaxation; on the other hand, much fine tuning is necessary toward improving the quality of the approximation, and this tuning depends both on the data and the number of iterations to be performed. However, the latter is quite unpredictable when one wants to find an approximation with a guaranteed quality. In brief, subgradient optimization is not adapted to finding an exact Lagrangian value. Here, we simply settle for an analysis of the approximation after 100 iterations.

The initial target value v^0 is obtained by Dixon and Silver's heuristic [19]. For the step size, we set $\lambda^0 = 2$, and halve λ if no improvement in the lower bound has occurred after 5 consecutive iterations. Several techniques have been proposed in the literature to update the target value v^0 of a multi-item capacitated lot-sizing problem: some smoothing procedures [11,55] and a primal restriction described next [53]. Given the value of y found at each subgradient iteration, we consider a primal restriction P_y of the problem P. Since any feasible solution of P_y is feasible for P, it may provide a better objective value than any solution found so far, and its objective value becomes the new target value. If the production periods are predetermined (by fixing y_{it} at 0 or 1 for $i = 1, \ldots, I$ and $t = 1, \ldots, T$), the restriction obtained is a network flow problem which can be solved very efficiently. Note that if the capacity absorptions a_i varied in time (a_{it}), the restricted problem would be a generalized network problem that can also be solved by efficient algorithms (see, for example, [34]). When a problem P has many products, relaxation methods for very large-scale networks are appropriate to solve P_y itself because there are only T sources (the productive resources in each period), but the number of demand nodes TI and the number of arcs increase rapidly. Also, changing the value of y only affects primal feasibility, so that a dual reoptimization algorithm may provide some computational advantages. Finally, the cost structure allows rather extensive streamlining: in the particular case of non-increasing production costs, a forward pass algorithm provides the optimum [28].

Following [16], we simply solve P_y as a transportation problem where some of the arcs may be forbidden to prevent backlogging or production in a period not selected:

$$\mathrm{P_y}: \quad v(P_y) = \min \sum_{i=1}^{I} \sum_{t=1}^{T} \sum_{\tau=1}^{T} \beta_{it\tau} \xi_{it\tau} \tag{51}$$

$$\text{subject to} \sum_{t=1}^{T} \xi_{it\tau} \geq a_i d'_{i\tau} \quad \text{for } i = 1, \ldots, I \text{ and } \tau = 1, \ldots, T, \tag{52}$$

$$\sum_{i=1}^{I} \sum_{\tau=1}^{T} \xi_{it\tau} \leq c_t \quad \text{for } t = 1, \ldots, T, \tag{53}$$

$$\xi_{it\tau} \geq 0 \quad \text{for } i = 1, \ldots, I \text{ and } t = 1, \ldots, T \text{ and } \tau = 1, \ldots, T, \tag{54}$$

where

$$\beta_{it\tau} = \begin{cases} (p_{it} + \sum_{q=t}^{\tau-1} h_{iq})/a_i, & \text{if } y_{it} = 1 \text{ and } t \leq \tau; \\ \infty & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, I$ and $t = 1, \ldots, T$ and $\tau = 1, \ldots, T$,

and $\xi_{it\tau}$ is the amount produced in period t to help satisfy the demand of product i in period τ (expressed in resource units).

The transportation problem is solved with an efficient primal simplex algorithm [51]. To determine whether P_y is feasible or not, we first perform a necessary but not sufficient test. If this test is satisfied, we solve the transportation problem P_y (which is infeasible in many cases). Note that the test does not guarantee that P_y will be feasible, but because of its simplicity, we use it as a convenient screening for feasibility. This will be indicated in theorem 12.

THEOREM 12

Given a set-up plan $y = (y_{it})$, $i = 1, \ldots, I$ and $t = 1, \ldots, T$, P_y is feasible only if

$$\sum_{t=1}^{k} \sum_{i=1}^{I} y_{it} a_i d_{it,n_y} \leq \sum_{t=1}^{k} c_t \quad \text{for } k = 1, \ldots, T, \tag{55}$$

where

$$n_y = \tau \text{ if } y_{it} = 1,\ y_{i,\tau+1} = 1 \text{ and } y_{ij} = 0, \quad \text{for } t < j \leq \tau,$$

with the same abusive notation: $y_{i,T+1} = 1$.

Proof

Given a set-up plan $y = (y_{it})$, $i = 1, \ldots, T$ and $t = 1, \ldots, T$, since all demands must be met before their due dates, any feasible solution of P_y is feasible for a multi-item capacitated lot-sizing problem Π with the same parameters and demands: $\delta_{it} = d_{it,n_y}$. The following necessary and sufficient condition for a feasible solution of a multi-item capacitated lot-sizing problem P to exist is given by Florian and Klein [26]:

$$\sum_{t=1}^{k} \sum_{i=1}^{I} a_i d'_{it} \leq \sum_{t=1}^{k} c_t \quad \text{for } k = 1, \ldots, T,$$

which, applied to Π, gives exactly (55). This proves that P_y must satisfy (55) to be feasible; however, even if Π satisfies this condition, a feasible solution of Π may not be feasible for P_y. □

Note that the quantities need to be calculated anyway to evaluate the subgradient direction for the next iteration. The test is therefore cheap and relatively effective.

To obtain the initial values of the Lagrangian multipliers, we solve the transportation problem P_y, where y is the solution obtained by Dixon and Silver's heuristic; then we use the dual variables corresponding to the resources as the initial values of the Lagrangian multipliers [53].

Our computational results show that for most of the problems tested, the subgradient optimization procedure does not perform as well as column generation. Because of the computational issues mentioned above, and the possibility of streamlining the search for a feasible value extensively, we have chosen not to include the computing time to solve the transportation problem, so as to put subgradient optimization in the most favorable light before comparison with column generation.

6.2. COLUMN GENERATION PROCEDURE

Our implementation closely follows [45]; we use an efficient version of Wagner and Whitin's algorithm [57].

We summarize the strategies that we have adopted.

- Dantzig and Van Slyke's generalized upper-bounding method [18];
- multiple pricing, which allows I columns (one for each subproblem) to be appended to the working basis;
- restricted master problem, which retains only the basic optimal columns between iterations;
- full series of pivots to reoptimize the master program before a new series of column generation.

Two implementations have also been tested for obtaining an initial basic solution:

- use the unit matrix corresponding to the slack variables for unused capacity, and artificial (key) variables for the convexity constraints (25);
- use uncapacitated solutions for the key columns, and slack or surplus capacity variables.

The superiority of the second implementation is particularly marked for the problems with relatively large capacities, in which an uncapacitated solution is rather

close to an optimal one. Computational experiments also point out the importance of a judicious scaling of the "big M" cost to drive artificial variables out of the basis: in the second implementation, setting large values perturbs the initial, uncapacitated solution too much and the optimization has to start from scratch. On the other hand, low values allow for smoother transition from uncapacitated to feasible solutions; however, they cannot be too small since they would not drive the artificial variables out of the basis. For the data set tested, a value between 50 and 500 is appropriate; 50 is adopted to report our results. These values can be viewed as a cost of overtime and readily give us some insight into the algorithmic behavior of the problem when overtime is allowed.

6.3. SHORTEST PATH FORMULATION

The matrix generator for P' is provided by Eppen and Martin, and makes minor departures from the basic formulation. Interested readers are referred to the original paper [22] for details. The problems are solved by first optimizing the uncapacitated version of the problem $P'_{32}(0)$ (each capacity c_t is multiplied by 10) and then reoptimized with the correct capacities. We have compared this approach with a straightforward one (optimization in one pass), and found that neither dominates the other. Our motivation for using the first approach (provided together with the matrix generator) is that the sizeable amount of time spent to generate an uncapacitated solution via linear programming could be saved by using a standard streamlined method such as Wagner and Whitin's algorithm. We do not implement this alternative, since its next natural enhancement is precisely the method of column generation that we want to evaluate, but we keep a separate record of the CPU time spent in each phase.

7. Description of test problems

The major objective of the experimental design is to investigate the ease of solution of various classes of data which are commonly viewed as replicating distinct real-world scenarios. In doing so, we closely follow the analysis of Maes and Van Wassenhove [46] conducted on heuristics for the multi-item capacitated lot-sizing problem. The additional advantage of using the same data characteristics is to give us insight as to whether these criteria are independent of the solution method.

7.1. PROBLEM SIZE

Corollary 2 asserts that the Lagrangian relaxation of the capacity constraints will yield values as close to the optimum as the number of items I is large. We want to validate this finding empirically, as well as to determine whether these better values are obtained at an increasing computing cost or not. Additionally, we need to verify that the number of periods T remains a true measure of problem size and determine the combined effect of I and T.

7.2. EFFECT OF INITIAL AND ENDING INVENTORY

The major application of a multi-item capacitated lot-sizing problem is in a dynamic production environment, where scheduling is often done on a "rolling horizon" basis. In each period, the plan for the current period is implemented and the plan for future periods remains tentative. The ending inventory of the current period becomes the initial inventory of the next period.

Virtually all test problems reported in the literature on the multi-item capacitated lot-sizing problem feature no initial inventories, with notable exceptions (Bahl and Ritzman [7], Graves [35]). With no initial inventories, the initial production will be devoted to at least as many products as are in demand in the first period. Also, the process can be perpetrated in future periods, resulting in a prolonged start-up phase clearly not representative of batching conditions. Billington [12] displays an example where the absence of initial inventory forces production lot for lot throughout the planning horizon. This example illustrates the predicament of schedulers reduced to stop-gap measures to perpetually catch up with the backlog of orders. The absence of initial inventory has particularly marked effects when the production capacity is tight, as our computational results confirm.

The lack of ending inventory will also distort the production plan, although in a less radical way. It is clear that the final inventory attenuates the boundary conditions artificially set by selecting a planning period. On the other hand, no prescriptive rule exists to determine its size in a dynamic environment; in our tests, we use a somewhat arbitrary value of one economic lot $EOQ_i = \sqrt{2s_i d_i / h_i}$ for each product i (d_i is the average demand). We also replicate the experiment with neither initial nor final inventory for ease of comparison with the results reported in other studies.

7.3. DATA GENERATOR

The experimental factors retained for the generation of data sets are essentially those selected by Maes and Van Wassenhove [46]: distribution of the magnitude and the timing of demand, level of capacity utilization (which we call *density*), capacity absorption variation, average time between orders $TBO_i = EOQ_i / d_i$, and initial and ending inventories adjustment described in the last section. The reader is referred to [46] for a discussion of the merit of the experimental design. We present the specific steps followed by the problem generator:

- First, two demand patterns are generated: *normal demand* in which a positive demand exists for each item in each period, and *lumpy demand* in which 25% of the periods have zero demand. The effect of demand lumpiness is described by Blackburn and Millen [15].
- The magnitude of the demand follows a normal distribution with mean (d_i) 100 and 125 for the normal and the lumpy case, respectively, and standard deviation (σ_i) selected from a uniform distribution with specified range.

- To analyze the potential effect of product dominance, each product is assigned a different capacity absorption coefficient (a_i), Clearly, a product with large absorption will have a greater impact on the schedule than if its use of capacities is small. The values of these coefficients are selected from a uniform distribution with predefined range.
- The production cost p_{it} is assumed to be uniform over all the periods. Because the given price of production must be paid no matter when the item is produced, it is entirely defined by the demand and does not affect the optimization. Accordingly, we disregard this portion of the total cost by setting $p_{it} = 0$ for all products and all periods.
- The holding cost h_{it} is normalized at 1. (Recall that we can handle product differences through various values of the capacity absorption (a_i).) The average time between orders (TBO_i) is generated from a uniform distribution with prespecified range. The value of the set-up cost s_i for each product is calculated using the economic order quantity:

$$s_i = \frac{h_i d_i TBO_i^2}{2}.$$

- The initial (*respectively*, final) inventories are based on TBO_i. They are handled indirectly by subtracting them from the initial demands (*respectively*, adding them to the final demands).
- The production capacities are derived from the average capacity requirement $\sum_{i=1}^{I} a_i d_{i1T}/T$ by dividing it by the *density* δ and adjusting it by a factor $0.9 + 0.2\rho_t$, where ρ_t is a random variable uniformly distributed between 0 and 1:

$$c_t = (0.9 + 0.2\rho_t)\frac{\sum_{i=1}^{I} a_i d_{i1T}}{T\delta}.$$

Unlike [46], we have not included a specific factor for increase of demand over time. However, the presence of initial inventories which reduce the initial demand has roughtly the same effect.

8. Computational analysis

The codes for subgradient optimization and column generation are compiled by a FORTVS compiler (optimize level 3) under CMS VM/SP 4.0 for an IBM 3081, whereas the shortest path formulation model is solved by LINDO (release March 1985), a commercial package, in the same environment.

8.1. THE PRIMAL SIMPLEX ALGORITHM VERSUS COLUMN GENERATION

In a preliminary test, we first compare column generation and shortest path formulation on a sample of 16 problems taken from [22]. Table 1 presents the CPU time required to reach the optimal value of the relaxation $v(P'_{32}) = v(P_3)$ with both methods; the CPU time for the simplex algorithm is decomposed into time to reach

Table 1

Problems from the literature: primal simplex algorithm for shortest path formulation $v(P'_{37})$ versus column generation $v(P_3) = v(P'_{37})$; computing times to reach optimal lower bound in CPU seconds (shortest path: initial solution time – final solution time)

Problem no.	Name	I	T	Shortest path	Column generation
1	TVW1	8	8	0.39–0.95	0.05
2	TVW2	8	8	0.39–0.24	0.02
3	TVW3	8	8	0.39–0.13	0.01
4	TVW4	8	8	0.39–0.07	0.00
5	TVW50	50	8	10.96–24.15	0.46
6	TVW100	100	8	37.19–44.65	1.23
7	TVW150	150	8	79.62–144.38	1.88
8	TVW200	200	8	137.33–88.28	2.30
9	DSBASE	20	13	7.30–0.77	0.15
10	DS115L	100	10	66.42–4.02	0.16
11	DS110L	100	10	66.37–6.01	0.26
12	DS105L	100	10	66.43–5.09	0.31
13	DS100M	100	10	66.42–8.72	0.55
14	DS199T	100	10	66.40–12.43	0.62
15	DS198T	100	10	66.41–18.21	0.65
16	DS200M	200	10	256.17–32.56	1.40

an uncapacitated solution and time to obtain $v(P'_{32})$ for the reasons explained in section 6.3. The CPU time needed to generate the matrix of the linear program P'_{32} is excluded, because we feel it could be shortened considerably with an efficient matrix generator, and because the matrix need not be re-generated in a typical post-optimal or side case analysis. Decomposing the CPU times allows us to draw a clear distinction between the data sets of [53] and its extension called TVW on one hand, and the data set DS based on [19] on the other hand. Because so much time is spent on solving the uncapacitated problem, a global CPU time would not enable us to see that the data set DS is much easier to solve than TVW. We attribute this to the sparse demand and relatively large capacity of DS, which leaves many production variables

zero at all iterations. For all problems, column generation is faster than the general-purpose simplex algorithm.

In a second set of experiments, reported in table 2, we analyze the impact of probem size on both algorithms. To limit the cost of the experiments, we only generate problems with initial and ending inventories. It will appear clearly in section 8.3 that among all other data characteristics, the density of the problem has

Table 2

Effect of size and density: primal simplex algorithm for shortest path formulation $v(P'_{37})$ versus column generation $v(P_3) = v(P'_{37})$; computing times to reach optimal lower bound in CPU seconds

Set	I	T	Density	Shortest path	Column generation
1	20	8	0.80	2.77–0.22	0.02
2	20	8	0.95	2.77–0.79	0.08
3	20	16	0.80	18.62–1.73	0.09
4	20	16	0.95	18.62–7.34	0.35
5	50	10	0.80	24.70–0.91	0.05
6	50	10	0.95	24.72–7.59	0.22
7	100	10	0.80	87.36–3.28	0.03
8	100	10	0.95	87.53–33.27	0.45

the largest impact on its difficulty and we include this third factor in the analysis. First, the number of products is fixed at 20, while the number of periods is successively set at 8, 16 and 32. Next, the number of products is set at 50, then 100, while the number of periods is fixed at 10. In both cases, we test two density levels: 0.8 and 0.95. For each set, five problem instances are generated. We report only the problems which we can solve within the size limitation of the linear programming package LINDO. (Note that the DS problems reported in table 1 are even larger, but the number of variables is not prohibitive in that case.)

For both algorithms, the CPU time is proportional to the number of products but increases sharply with the number of periods. With conclusive evidence that column generation yields solutions faster than a general-purpose simplex algorithm, we omit the latter from subsequent experiments.

8.2. EFFECT OF PROBLEM SIZE

To investigate more systematically the effect of problem size (number of items and number of periods) and density on its difficulty, we solve a series of 480 problems by column generation and subgradient optimization. The experimental design used for these experiments is summarized in table 3. The demand of 75% of the products is assumed to be normal, and the rest to be lumpy. The number of products ranges from 10 to 80 and the number of periods from 6 to 24. We also consider the influence

Table 3

Experimental design: effect of problem size; column generation versus subgradient optimization

Parameter		Values and distribution
Average demand	d_i	100 (125*), normal
Standard deviation of demand	σ_i	[10, 50], uniform
Demand type		75% normal, 25% lumpy
Capacity absorption	a_i	[1, 5], uniform
Average time between orders	TBO_i	[1, 4], uniform
Holding cost	h_i	1, fixed
Capacity density	δ	0.80 (0.85*), 0.90, 0.95
Number of products	I	10, 20, 40, 80
Number of periods	T	6, 12, 18, 24

*Parameters for lumpy demand.
*Parameters for the data sets that include inventories.

of initial and final inventories by generating one set of cases with inventory, and replicating the experiment with another set featuring no inventory. Three levels of density are tested: 0.80, 0.90 and 0.95; however, with a density of 0.80 in the presence of inventory, virtually all solutions are uncapacitated; therefore, in this case we raise the density to 0.85. For each set, five problem instances are generated.

Table 4 shows the results of column generation. The first number in each column is the average CPU time, while the average number of iterations is shown in parentheses.

- The number of periods has the most conspicuous effect on the difficulty of the problem, as the computational time required to solve the Lagrangian relaxation of these problems increases superlinearly as a function of the number of periods. This increase is essentially due to the growing burden of solving the linear program for the master problem (the basis of which increases proportionally with the number of periods).
- The initial and ending inventories have a strong impact on the difficulty of the problem, which confirms the observation made in section 7.2. The joint effect of the number of periods and the inventory factor is a direct consequence of each of their marginal effects.
- The effect of problem density resembles that of inventories (for example, it affects the performance of the algorithm with respect to the number of periods in a similar way).
- Within the range of items tested here, the influence of the number of items on the difficulty of the problem, measured in CPU time, varies greatly. In

Table 4

Effect of problem size, density and inventory: column generation $v(P_3)$; CPU seconds (number of iterations) to reach optimal lower bound

			T							
Inv	Density	I	6		12		18		24	
		10	0.022	(10.0)	0.078	(10.0)	0.178	(15.0)	0.354	(18.8)
		20	0.048	(9.0)	0.184	(13.8)	0.298	(14.2)	0.586	(17.0)
	0.80	40	0.110	(9.0)	0.334	(14.0)	0.596	(13.6)	1.340	(17.0)
		80	0.456	(13.6)	0.828	(13.0)	1.390	(13.6)	2.268	(14.4)
		10	0.038	(15.4)	0.166	(27.0)	0.460	(41.2)	0.922	(50.8)
		20	0.100	(18.4)	0.438	(32.4)	0.684	(29.6)	1.538	(37.2)
No	0.90	40	0.212	(17.6)	0.756	(25.2)	1.842	(31.4)	3.440	(36.4)
		80	0.698	(20.4)	2.288	(28.8)	4.084	(27.8)	5.374	(24.6)
		10	0.050	(17.8)	0.270	(50.6)	0.892	(96.2)	1.618	(77.2)
		20	0.098	(17.6)	0.662	(58.4)	1.508	(78.6)	3.878	(121.8)
	0.95	40	0.310	(25.4)	1.814	(80.8)	4.396	(103.4)	7.040	(84.4)
		80	0.998	(32.0)	4.804	(81.4)	9.332	(85.2)	21.468	(139.8)
		10	0.012	(5.4)	0.034	(8.0)	0.096	(10.0)	0.180	(12.4)
		20	0.012	(4.2)	0.042	(6.0)	0.118	(8.6)	0.166	(7.8)
	0.85	40	0.016	(2.4)	0.074	(4.8)	0.156	(7.4)	0.236	(6.6)
		80	0.022	(2.0)	0.086	(3.4)	0.116	(3.2)	0.314	(6.0)
		10	0.016	(8.0)	0.068	(11.8)	0.212	(20.2)	0.450	(25.4)
		20	0.028	(6.8)	0.098	(10.2)	0.346	(17.0)	0.666	(20.0)
Yes	0.90	40	0.052	(7.0)	0.148	(8.2)	0.332	(10.4)	0.660	(11.4)
		80	0.122	(7.8)	0.234	(7.0)	0.648	(10.8)	1.464	(13.4)
		10	0.020	(9.8)	0.106	(17.6)	0.372	(32.6)	1.438	(89.2)
		20	0.036	(7.8)	0.188	(17.4)	0.368	(16.6)	1.144	(35.0)
	0.95	40	0.068	(8.2)	0.256	(11.2)	0.772	(16.8)	2.196	(33.6)
		80	0.124	(7.2)	0.682	(14.4)	1.928	(20.8)	3.448	(25.0)

the presence of inventories, the increase of CPU time is less than proportional to the number of items, and may possibly reach a limit (notice that the number of iterations decreases steadily). However, in the absence of inventories, the number of iterations remains approximately constant, and the CPU time increases roughly linearly with the number of items.

We now turn to the results obtained by subgradient optimization. In table 5, we first report some average CPU times to implement 100 iterations of the algorithm as a function of problem size.

Table 5

Effect of problem size: subgradient optimization $v(P_3)$; average CPU time for 100 iterations

$I \backslash T$	6	12	18	24
10	0.21	0.30	0.43	0.58
20	0.28	0.54	0.79	1.09
40	0.53	1.00	1.43	2.07
80	0.99	1.98	3.13	4.27

Table 6

Effect of problem size, density and inventory: subgradient optimization $v(P_3)$; relative difference between subgradient optimization and optimal lower bounds in % (number of problems reaching 1% out of 5)

			T							
Inv	Density	I	6		12		18		24	
		10	0.06	(5)	0.09	(5)	0.10	(5)	0.02	(5)
		20	0.03	(5)	0.04	(5)	0.08	(5)	0.02	(5)
	0.80	40	0.02	(5)	0.03	(5)	0.04	(5)	0.04	(5)
		80	0.04	(5)	0.02	(5)	0.01	(5)	0.02	(5)
		10	0.49	(4)	0.90	(4)	1.44	(4)	0.34	(4)
		20	0.38	(4)	0.63	(4)	0.43	(4)	0.41	(4)
No	0.90	40	0.36	(5)	0.57	(4)	0.34	(5)	0.29	(5)
		80	0.18	(5)	0.59	(4)	0.35	(5)	0.82	(4)
		10	0.93	(2)	2.07	(2)	2.09	(3)	2.16	(3)
		20	0.18	(5)	2.25	(0)	2.03	(1)	1.83	(3)
	0.95	40	0.53	(4)	2.75	(1)	2.42	(0)	1.65	(2)
		80	0.50	(4)	2.52	(2)	1.76	(0)	3.24	(0)
		10	0.02	(5)	0.02	(5)	0.02	(5)	0.02	(5)
		20	0.00	(5)	0.00	(5)	0.01	(5)	0.00	(5)
	0.85	40	0.00	(5)	0.00	(5)	0.00	(5)	0.00	(5)
		80	0.00	(5)	0.00	(5)	0.00	(5)	0.00	(5)
		10	0.00	(5)	0.09	(5)	0.05	(5)	0.14	(5)
		20	0.01	(5)	0.03	(5)	0.01	(5)	0.01	(5)
Yes	0.90	40	0.00	(5)	0.00	(5)	0.00	(5)	0.00	(5)
		80	0.00	(5)	0.00	(5)	0.00	(5)	0.00	(5)
		10	0.03	(5)	0.10	(5)	0.65	(4)	0.39	(5)
		20	0.02	(5)	0.05	(5)	0.04	(5)	0.08	(5)
	0.95	40	0.00	(5)	0.01	(5)	0.02	(5)	0.01	(5)
		80	0.01	(5)	0.01	(5)	0.03	(5)	0.01	(5)

We do not report CPU times for every entry of table 6 because they are roughly the same for all the problems of a same size (regardless of the density and inventory). To solve 100 iterations, the time taken by subgradient optimization is approximately

proportional to number of items and number of periods. Note that, with our efficient implementation of Wagner and Whitin's algorithm, the computing time grows linearly with $TBO * T$. The CPU times are generally higher than their counterparts recorded in table 4, except for the data sets with a large number of periods, high density and no inventory (even in this case, remember that the time to solve the transportation problem in the subgradient algorithm is not included). For those problems, subgradient optimization does not reach a good Lagrangian bound.

Table 6 shows the results of subgradient optimization. The first number in each column is the percentage difference between the Lagrangian value $v(P_3)$ and its lower estimate provided by the algorithm. In parentheses is the number of problems (out of 5) which reach 1% relative difference within 100 iterations. This measure is admittedly much less accurate than that of table 4, and to compare the two tables, we have to use the following analogy: the more difficult the problem (in terms of CPU time for column generation), the higher the percentage difference (therefore the smaller the number of problems reaching 1%).

- The initial and ending inventories have also a strong impact on the behavior of subgradient optimization (as is the case with column generation).
- The effect of the problem density is much more pronounced in the absence of inventories, but affects the performance of the algorithm in a similar way as column generation.
- Within the range of items tested here, the influence of the number of items does not display a systematic pattern.

In most problems, it takes less time for the column generation to reach the Lagrangian optimum $v(P_3)$ than for the subgradient optimization to approximate this value. It can be claimed that the size of the problems investigated is not large enough to allow subgradient optimization to show an advantage. However, our choice of a number of periods is dictated by practical considerations: very few planners perform long-range analyses over 18 or 24 periods, and a more detailed (small bucket) planning would warrant a different model altogether. There is also general agreement that the last fractions of accuracy in the objective function are gained at increasing computational costs. Even for problems where subgradient optimization shows shorter times, the quality of the lower bound rarely reaches 1%. Our experience allows us to safely conjecture that the subgradient algorithm would take a much longer time to obtain an accuracy comparable with that of column generation. In summary, subgradient optimization seems to be attractive only when a rough estimate of the Lagrangian optimum is necessary; additional experiments not reported here also indicate that the quality of this lower bound has a crucial effect on the efficiency of enumerative methods to find the exact solution of the multi-item capacitated lot-sizing problem.

8.3. EFFECT OF DATA PATTERNS

This experimental design tries to capture the effect of data patterns on both subgradient optimization and column generation. Its parameters are summarized in table 7, where the problem size is fixed at a median value $I = 20$, $T = 12$.

Table 7

Experimental design: effect of data patterns; column generation versus subgradient optimization

Parameter	Values
Standard deviation of demand	High [0, 60], medium [0, 30], low [0, 10]
Demand type	Normal, lumpy
Capacity absorption	Constant: 1, random [1, 5]
Average time between orders	High [1, 6], low [1, 2]
Capacity density	High: 0.95, medium: 0.90, low: 0.80 (0.85*)

*Parameter for the data sets that include inventories.

The problems are generated as described in the previous section. A complete factorial design across the five parameters is adopted, making up 72 problem classes. For each problem class, two types of problems – with and without initial and ending inventory – are tested with 5 replicates, which yields a total of 720 problems. A first analysis of the results revealed that demand variability (σ_i) and capacity absorption (a_i) did not significantly affect the computational time. Therefore, for both sets, the results presented below are aggregated into 12 problem classes, each containing 30 problem instances. To show the level of difficulty of each problem class, the results are summarized in tables 8 and 9.

Table 8

Effect of data patterns: column generation $v(P_3)$; CPU seconds and (number of iterations) to reach optimal lower bound

		Low *TBO*				High *TBO*			
Inv	Density	Normal		Lumpy		Normal		Lumpy	
No	0.80	0.064	(6.0)	0.044	(5.6)	0.313	(21.1)	0.213	(15.0)
	0.90	0.150	(12.8)	0.117	(10.8)	0.642	(55.8)	0.423	(35.9)
	0.95	0.277	(26.5)	0.227	(19.9)	0.867	(84.4)	0.523	(49.8)
Yes	0.85	0.037	(4.6)	0.053	(6.5)	0.084	(8.4)	0.084	(8.6)
	0.90	0.077	(7.4)	0.104	(9.7)	0.103	(10.3)	0.113	(11.0)
	0.95	0.164	(15.3)	0.166	(14.5)	0.164	(14.4)	0.175	(16.5)

Table 8 shows the results for column generation; as in table 4, the first number in each column is the average CPU time, and the number of iterations is shown in parentheses

- The presence of initial and ending inventories has the largest impact on the difficulty of the problem.
- When *TBO* is low, the optimal solution will resemble lot-for-lot scheduling, and the problem is relatively easy to solve.
- The effect of problem density is much more pronounced in the absence of inventories.
- Demand lumpiness affects the difficulty of the problem significantly only in the absence of inventories.

Table 9

Effect of data patterns: subgradient optimization $v(P_3)$; relative difference between subgradient optimization and optimal lower bound in % (number of problems reaching 1% out of 30)

		Low *TBO*				High *TBO*			
Inv	Density	Normal		Lumpy		Normal		Lumpy	
	0.80	0.00	(30)	0.00	(30)	0.19	(30)	0.17	(29)
No	0.90	0.07	(30)	0.02	(30)	2.13	(6)	0.92	(20)
	0.95	0.16	(29)	0.07	(30)	5.61	(1)	2.60	(5)
	0.85	0.00	(30)	0.02	(30)	0.01	(30)	0.01	(30)
Yes	0.90	0.00	(30)	0.01	(30)	0.02	(30)	0.02	(30)
	0.95	0.15	(29)	0.13	(29)	0.10	(30)	0.17	(28)

Table 9 shows the results of subgradient optimization. The first number in each column is the percentage difference between the Lagragian value $v(P_3)$ and its lower estimate provided by the algorithm. In parentheses is the number of problems (out of 30) which reach 1% relative difference within 100 iterations. To compare tables 8 and 9, we use the same analogy as in section 8.2: the more difficult the problem (in terms of CPU time for column generation), the higher the percentage difference (so the smaller the number of problems reaching 1%). The CPU time to implement 100 iterations is affected mainly by *TBO*. For low *TBO*, we find an average of 0.47 seconds, whereas for high *TBO*, we find an average of 0.54 seconds (excluding the transportation problem). Demand lumpiness increases the CPU times slightly. These times are generally higher than most CPU times for column generation. All the observations from table 8 apply to table 9 as well.

9. Conclusions

Our effort to determine an efficient solution approach for the multi-item capacitated lot-sizing problem has led us naturally to consider the three traditional areas of problem solving:

- At the *methodological* level, our computational complexity result re-establishes the desirability of exact optimization methods for which we sort out good relaxations from bad ones; measuring their quality ex-ante has saved us from implementing and then rejecting some algorithms, as could have happened if we had used empirical testing only. Further relaxation analysis can be found in [54].
- Among various *solution strategies*, we have pointed out the advantage of column generation, which in our case does not display a reputed slow convergence; moreover, column generation is a good candidate for vector processing, because a major portion of the computation is devoted to solving a dense linear program. Eppen and Martin have shown that the new formulation is very practical because it can readily be implemented with a general purpose linear programming package; to their finding, we add that the linear program leads to very efficient customized implementations, one of them being column generation (a few codes are already available).
- At the *implementation* level, we have shown that problem difficulty can be characterized by some parameters which reflect directly the manufacturing environment. Realistic problems with initial inventories and relatively ample capacity are much easier to solve than some data sets proposed in the past.

Our focus on relaxation or dual methods must be set against the vast body of literature on primal heuristics. By using the same experimental design, we are able to detect a clear symmetry between the performance of primal and dual methods, which allows us to qualify certain data sets as intrinsically difficult (i.e. high density, few items, many periods) or easy.

It is not a coincidence that column generation and subgradient optimization span two formulations of the same problem; the adequacy of a same data structure to embody two different algebraic representations extends to other logistic problems and has important ramifications in modeling. For instance, what is the similarity between the polyhedral structure of the two formulations, and its implications on the relationship and the relative power of cutting planes of either formulation? As a practical issue, would the addition of cutting planes to the master problem preserve the benefit of a partitioned approach initially derived from the small number of capacity constraints?

Acknowledgements

We would like to thank Gary Eppen and Kipp Martin for providing us with a copy of the data and user subroutines to generate the linear programs, Luk Van Wassenhove and Johan Maes for the problem generator, V. Srinivasan and Gerald Thompson for portions of the codes used in this study, and Harish Bahl for some valuable suggestions on column generation.

The problem generator, the subgradient and column generation codes are available from the authors.

References

[1] C.H. Aikens, The optimal design of a physical distribution system on a multicommodity multi-echelon network, Ph.D. Thesis, The University of Tennessee, Knoxville, TN (1982).
[2] C.H. Aikens, Facility location models for distribution planning, Eur. J. Oper. Res. 22(1985)263–279.
[3] A.I. Ali, R.V. Helgason, J.L. Kennington and H. Hall, Computational comparison among three multicommodity network flow algorithms, Oper. Res. 23(1980)995–1000.
[4] A.I. Ali, D. Barnett, K. Farhangian, J. Kennington, B. McCarl, B. Patty, B. Shetty and P. Wong, Multicommodity network problems: Applications and computations, IIE Trans. 16 (2) (1984)127–134.
[5] D. Atkins, A survey of lower bounding methodologies for production/inventory model, Ann. Oper. Res., this volume.
[6] H.C. Bahl, Column generation based heuristic algorithm for multi-item scheduling, AIIE Trans. 15 (2) (1983)136–141.
[7] H.C. Bahl and L.P. Ritzman, A cyclical scheduling heuristic for lot sizing with capacity constraints, Int. J. Prod. Res. 22(1984)791–800.
[8] I. Barany, T.J. Van Roy and L.A. Wolsey, Strong formulations for multi-item capacitated lot sizing, Manag. Sci. 30 (10) (1984)1255–1261.
[9] D.P. Bertsekas and P. Tseng, Relaxation methods for minimum cost network flow problems, Technical Report LIDS-P-1339, MIT, Cambridge, MA (1983).
[10] D.P. Bertsekas and E.M. Gafni, Projected Newton methods and optimization of multicommodity flows, IEEE Trans. Automatic Control AC28 (12) (1983)1090–1096.
[11] P.J. Billington, Multi-level lot-sizing with a bottleneck work center, Ph.D. Thesis, Graduate School of Business and Public Administration, Cornell University (1983).
[12] P.J. Billington, The capacitated lot-sizing problem (CLSP), review and extensions, Working Paper No. 84-13, College of Business Administration, Northeastern University (1984).
[13] G.R. Bitran and H.H. Yanasse, Computational complexity of the capacitated lot size problem, Manag. Sci. 28(1982)1174–1186.
[14] G.R. Bitran and H. Matsuo, The multi-item capacitated lot size problem: Error bounds of Manne's formulations, Manag. Sci. 32(1986)350–359.
[15] J.D. Blackburn and R.A. Millen, Heuristic lot-sizing performance in a rolling-schedule environment, Decision Sci. 11(1980)691–701.
[16] E.H. Bowman, Production scheduling by the transportation method of linear programming, Oper. Res. 4(1956)100–103.
[17] G. Cornuéjols, M.L. Fisher and G.L. Nemhauser, Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms, Manag. Sci. 23(1977)789–810.
[18] G.B. Dantzig and R.M. Van Slyke, Generalized upper bounding techniques, J. Comp. Syst. Sci. 1(1967)213–226.

[19] P. Dixon and E.A. Silver, A heuristic solution procedure for the multi-item, single-level, limited capacity, lot-sizing problem, J. Oper. Management 2 (1) (1981)23–39.
[20] A. Dogramaci, J.C. Panayiotopoulos and N.R. Adam, The dynamic lot-sizing problem for multiple items under limited capacity, AIIE Trans. 13 (4) (1981)294–303.
[21] B.P. Dzielinski and R.E. Gomory, Optimal programming of lot sizes, inventories, and labor allocations, Manag. Sci. 11 (9) (1965)874–890.
[22] G.D. Eppen and R.K. Martin, Solving multi-item capacitated lot-sizing problems using variable redefinition, Oper. Res. 35 (6) (1987)832–848.
[23] J.R. Evans, Network-based optimization algorithms for the capacitated multi-item lot sizing problem, Comp. Ind. Eng. 9 (3) (1985)297–305.
[24] A. Federgruen, A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time, Graduate School of Business, Columbia University (1989).
[25] M. Florian, J.K. Lenstra and A.H.G. Rinnooy Kan, Deterministic production planning: Algorithms and complexity, Manag. Sci. 26 (7) (1980)669–679.
[26] M. Florian and M. Klein, Deterministic production planning with concave costs and capacity constraints, Manag. Sci. 18 (1) (1971)12–20.
[27] M. Frank and P. Wolfe, An algorithm for quadratic programming, Naval Res. Logist. Quart. 3 (2) (1956)95–110.
[28] H. Gabbay, Multi-stage production planning, Manag. Sci. 25(1979)1138–1148.
[29] M.R. Garey and D.S. Johnson, Strong NP-completeness results: Motivation, examples and implications, J. ACM 25(1978)499–508.
[30] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).
[31] A.M. Geoffrion, Lagrangian relaxation for integer programming, Math. Progr. Study 2(1974)82–114.
[32] A.M. Geoffrion and G.W. Graves, Multicommodity distribution system design by Benders decomposition, Manag. Sci. 20 (5) (1974)822–844.
[33] A.M. Geoffrion, Better distribution planning with computer models, Harvard Business Rev. 54 (4) (1976)92–99.
[34] F. Glover, J. Hultz, D. Klingman and J. Stutz, Generalized networks: A fundamental computer-based planning tool, Manag. Sci. 24 (12) (1978)1209–1220.
[35] S.C. Graves, Using Lagrangian techniques to solve hierarchical production planning systems, Manag. Sci. 28 (3) (1982)260–275.
[36] M. Guignard and K. Opaswongkarn, Primal and dual approaches to single- and multi-commodity capacitated plant location problems, Technical Report No. 53, Department of Statistics, University of Pennsylvania (1983).
[37] D.W. Hearn, S. Lawphongpanich and J.A. Ventura, Finiteness in restricted simplicial decomposition, Oper. Res. Lett. 4 (3) (1985)125–130.
[38] M. Held, P. Wolfe and H.P. Crowder, Validation of subgradient optimization, Math. Progr. 6 (1) (1974)62–88.
[39] C.A. Holloway, An extension of the Frank and Wolfe method of feasible directions, Math. Progr. 6(1974)14–27.
[40] J.L. Kennington, A survey of linear cost multicommodity network flows, Oper. Res. 26(1978)209–236.
[41] J.L. Kennington and R.V. Helgason, *Algorithms for Network Programming* (Wiley, New York, 1980).
[42] P.R. Kleindorfer and E.F.P. Newson, A lower bounding structure for lot-sizing scheduling problems, Oper. Res. 23 (2) (1975)299–311.
[43] J.G. Klincewicz, A large-scale distribution and location model, AT&T Tech. J. 64 (7) (1985) 1705–1730.
[44] M.R. Lambrecht and H. Vanderveken, Heuristic procedures for the single operation, multi-item loading problem, AIIE Trans. 11 (4) (1979)319–326.

[45] L.S. Lasdon and R.C. Terjung, An efficient algorithm for multi-item scheduling, Oper. Res. 19 (4) (1971)946–969.
[46] J. Maes and L.N. Van Wassenhove, Multi item single level capacitated dynamic lotsizing heuristics: A computational comparison (Part I: Static case), IIE Trans. (1986)114–123.
[47] A.S. Manne, Programming of economic lot sizes, Manag. Sci. 4 (2) (1958)115–135.
[48] R.K. Martin, Generating alternative mixed integer programming models using variable redefinition, Oper. Res. 35 (6) (1987)820–831.
[49] Y. Pochet and L.A. Wolsey, Lot-size models with backlogging: Strong reformulations and cutting planes, Technical Report, Université Catholique de Louvain, Belgium (1986).
[50] Y.-S. Shan, A dynamic multicommodity network flow model for real time optimal rail freight car management, Ph.D. Thesis, Civil Engineering Department, Princeton University, Princeton, NJ (1985).
[51] V. Srinivasan and G.L. Thompson, Benefit-cost analysis of coding techniques for the primal transportation algorithm, J. ACM 20(1973)194–213.
[52] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions I*, Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen, Band 163 (Springer, New York, 1970).
[53] J.-M. Thizy and L.N. Van Wassenhove, Lagrangian relaxation for the multi-item capacitated lot-sizing: A heuristic implementation, IIE Trans. 17 (4) (1985)308–313.
[54] J.-M. Thizy, Analysis of Lagrangian decomposition for the multi-item capacitated lot-sizing problem, INFOR, to appear.
[55] W.W. Trigeiro, A dual-based heuristic for the capacitated lot-sizing problem, Ph.D. Thesis, Graduate School of Business and Public Administration, Cornell University (1985).
[56] B. Von Hohenbalken, Simplicial decomposition in nonlinear programming algorithms, Math. Progr. 13(1977)49–68.
[57] H.M. Wagner and T. Whitin, Dynamic version of the economic lot size model, Manag. Sci. 5 (1) (1958)89–96.
[58] W.I. Zangwill, A backlogging model and a multi-echelon model of a dynamic economic lot size production system: A network approach, Manag. Sci. 15 (9) (1969)506–527.