MANAGEMENT SCIENCE
Vol. 11, No. 9, July, 1965
Printed in U.S.A.

OPTIMAL PROGRAMMING OF LOT SIZES, INVENTORY AND LABOR ALLOCATIONS*

BERNARD P. DZIELINSKI AND RALPH E. GOMORY

IBM Corporation, Yorktown Heights, New York

The economic lot size programming problem as studied originally by A. S. Manne and later by B. P. Dzielinski, C. T. Baker and A. S. Manne, is the problem of making economic lot size, inventory and work force decisions in a multi-production process. When several thousand distinct items are involved, the large number of equations that result from the linear programming formulation makes computation infeasible. Also, a large number of variables are involved because of inclusion of alternative set-up sequences for each item. In this paper, the application of the Dantzig and Wolfe decomposition principle and a method for creating alternative set-up sequences as they are needed by means of a computation of the Wagner and Whitin type is described as a method for overcoming the computational difficulty.

A digital computer program has been developed using these methods. The results of some experiments where production was planned for a large number of distinct items are described.

Introduction

An application of the linear programming method to the problem of making economic lot-size decisions for a multi-item production process has been studied and reported on by A. S. Manne [14]. Recently, B. P. Dzielinski, C. T. Baker, and A. S. Manne made additional studies for making economic lot-size, inventory, and labor-force employment decisions by the use of the economic lot-size model. These studies indicate that linear programming offers a promising method for the practical economic planning of such activities [5].

The economic lot-size model developed for performing these tasks, however, contains certain characteristics which lead to difficulties when applications to large scale production situations are considered.

(i) First of all, a very large-sized linear programming problem results when lot-size decisions on several thousands of distinct items are considered. The model involves a large number of equations and puts computation beyond the capabilities of current linear programming codes. The difficulty may be overcome by performing certain approximations (by forming aggregate classes of items), but it may persist even with these approximations.

(ii) Secondly, the model contains an activity vector for every time-phased vector of labor-hour coefficients. These coefficients satisfy the demand requirements over time for an item. This results in an unmanageable number of variables, since in general, there will be a great many such feasible labor-hour coefficient vectors for each item.

* Received May 1964.

The purpose of this paper is to show how these objections can be met: first, by employing the methods of the Dantzig and Wolfe Decomposition [3], here the task is to accomplish a reduction in the large number of constraints in the original problem, second, by showing how the Simplex Algorithm can be augmented with the ideas of Ford and Fulkerson [6] to deal with the number of variables. This involves using information (the prices) generated by the Simplex Algorithm to simply create a new improved alternative labor-input coefficient vector. In this computation the Wagner and Whitin algorithm [17] replaces the method of looking over a vast collection of existing vectors as is required for the economic lot-size model. The approach and notation are closely related to [10].

The Problem

The problem studied here is given in Manne [14] and will be restated in our notation in equations (4.0)–(4.3) below. In this formulation an important part is played by the labor hour input coefficient vector. This is the vector that over time allocates the needed labor resources to the requirements for a particular item. These vectors, which are columns in the LP model, are obtained as follows: A T-period set-up sequence is denoted by a vector Δ_j and consists of components $\delta_{j\tau}$ taking on values of zero or one; i.e., when $\delta_{j\tau} = 1$, a setup is incurred in a period τ, otherwise no setup is incurred.

$$\Delta_j = \begin{bmatrix} \delta_{j1} \\ \vdots \\ \delta_{j\tau} \\ \vdots \\ \delta_{jT} \end{bmatrix} \qquad j = 1, \cdots, J$$

The demand requirements r_i for each item i in period τ, are denoted by $r_{i\tau}$. Corresponding to each of the Δ_j vectors, a time-phased production vector X_{ij} may be written as:

$$X_{ij} = \begin{bmatrix} x_{ij1} \\ \vdots \\ x_{ij\tau} \\ \vdots \\ x_{ijT} \end{bmatrix} \qquad \begin{array}{l} j = 1, \cdots, J \\ i = 1, \cdots, I \end{array}$$

The output levels, $x_{ij\tau}$, are determined by a set of rules; these rules essentially are equivalent to the rule, "each delivery requirement is satisfied out of production during the *nearest* preceding period in which a setup is incurred." It can be shown [5], [14] that only these outputs need be considered. The vector X_{ij} is denoted as a *production schedule.*

Specifically, when generated by the set-up plan Δ_j, the corresponding vector X_{ij}, must be feasible from the viewpoint of the delivery requirements; that is, the components of X_{ij} must satisfy

$$\sum_{\tau=1}^{t} x_{ij\tau} \geqq \sum_{\tau=1}^{t} r_{i\tau}, \qquad t = 1, \cdots, T-1, \tag{1}$$

and

$$\sum_{\tau=1}^{t} x_{ij\tau} = \sum_{\tau=1}^{T} r_{i\tau}, \tag{2}$$

For example, let us consider the case of three periods with requirements r_{i1}, r_{i2}, and r_{i3} for the first, second and third periods, respectively. The dominant feasible production schedules, that is the ones produced by the above rule are listed, and they are:

Sequence No.	Amount Produced in Period: τ		
	1	2	3
$j = 1$	$x = r_{i1} + r_{i2} + r_{i3}$	$x = 0$	$x = 0$
$j = 2$	$x = r_{i1} + r_{i2}$	$x = 0$	$x = r_{i3}$
$j = 3$	$x = r_{i1}$	$x = r_{i2} + r_{i3}$	$x = 0$
$j = 4$	$x = r_{i1}$	$x = r_{i2}$	$x = r_{i3}$

It is clear that no requirements are split in this process; that is, each $r_{i\tau}$ ($\tau = 1, 2, \cdots, T$) is allocated in its entirety to one period. The maximum number of such dominant production schedules is: $J = 2^{T-1}$. In a production process, it is possible to generate the corresponding labor-hour input coefficients, $l_{ijk\tau}$, the hours of each labor of type k needed for the production quantity, $x_{ij\tau}$, for each item i: ($i = 1, \cdots, I$; $k = 1, \cdots, K$) as:

$$l_{ijk\tau} = \begin{Bmatrix} 0 \\ \\ a_{ik} + b_{ik} \cdot x_{ij\tau} \end{Bmatrix} \quad \text{according, as:} \quad x_{ij\tau} \begin{matrix} = \\ \\ > \end{matrix} \begin{Bmatrix} 0 \\ \\ 0 \end{Bmatrix} \tag{3}$$

where a_{ij} and b_{ik} refer, respectively, to the labor set-up hours and labor production hours per unit of item i using labor type k.

Then, the corresponding *labor-hour input-coefficient vector* is denoted on opposite page.

The most economic production schedules, when determined by the linear programming computation, pick out at least one L_{ij} vector for each item, such that the set of vectors picked for all the items concurrently will optimize the defined objective function, subject to the restrictions of the problem.

The economic lot-size and work-force planning problem can be written as

$$\text{Minimize} \quad \sum_i \sum_j C_{ij}\,\theta_{ij} \tag{4.0}$$

subject to:

$$\sum_{ij} l_{ijk\tau}\theta_{ij} \leqq \sum_r H_k^{\tau} W_{k\tau}^r, \quad k = 1, \cdots, K, \ \tau = 1, \cdots, T, \tag{4.1}$$

$$\sum_j \theta_{ij} = 1, \qquad i = 1, \cdots, I, \tag{4.2}$$

$$L_{ij} = \begin{bmatrix} l_{ij11} \\ \cdot \\ \cdot \\ l_{ij1\tau} \\ \cdot \\ \cdot \\ l_{ij1T} \\ \cdot \\ \cdot \\ l_{ijk1} \\ \cdot \\ \cdot \\ l_{ijk\tau} \\ \cdot \\ \cdot \\ l_{ijkT} \\ \cdot \\ \cdot \\ l_{ijK1} \\ \cdot \\ \cdot \\ l_{ijK\tau} \\ \cdot \\ \cdot \\ l_{ijKT} \end{bmatrix} \qquad \begin{array}{l} j = 1, \cdots, J \\ i = 1, \cdots, I. \end{array}$$

and

(4.3) $$\theta_{ij} \geqq 0.$$

The variables θ_{ij} require some interpretation. There is one for each feasible labor-hour coefficient vector. If the θ_{ij} are integers, conditions (4.2) and (4.3) assure that they can take on only the values 0 or 1, and, in fact, for each i exactly one of the θ_{ij} will have value 1; the others will be zero. Thus, if the θ_{ij} are integers and satisfy (4.2) and (4.3), they can be interpreted as picking out from the existing list of feasible labor-hour coefficient vectors for each item, exactly one production schedule that is to be followed. To each θ_{ij} there corresponds a cost coefficient:

$$C_{ij} = \sum_{\tau=1}^{T} c_{i\tau} \cdot x_{ij\tau}$$

where $c_{i\tau}$ is a cost value indicating the discounted material cost of item i.

With this interpretation of the θ_{ij}, then (4.1) are the equations of labor balance in planning the production for the I items. We have the labor required on the left, in labor man-hours. The right hand side term gives the labor available $\sum_r H_k^r \cdot W_{k\tau}^r$ ($r = 1, \cdots, n$). The number of workers of the kth labor class and the rth payment type is $W_{k\tau}^r$. For example, $r = 1$ indicates straight time work; $r = 2$ indicates straight and overtime work. The number of hours a type r, k worker will work in a period is H_k^r. The problem is to find θ_{ij} that satisfy equations (4.1)–(4.3) and minimize (4.0).

Later we will describe a version of the model in which the $W_{k\tau}^r$ are variables. There will be costs $R_{k\tau}^r$ for using the labor of the various classes, and there will be equations connecting labor quantities in one period with labor quantities in the next, see (12.0)–(12.4) below. To simplify the exposition however we confine ourselves to a simpler model in this part.

Quite aside from the problem of getting integer θ_{ij}, we have in (4.0)–(4.3) a formidable linear programming problem. There are $I + KT$ rows. K and T may each reasonably be around 10 for many situations; I, the number of distinct items, may be from a few hundred to several thousand. When the number of items is much over one thousand, it becomes impossible to compute with the Manne model. Coupled with this, the number of columns can become overwhelming even for smaller I. For instance, if $T = 10$, then for each item the total number of feasible production vectors equal $J = 2^{10-1} = 512$. Thus, with $I = 500$, there would be at least 256,000 columns in the tableau. It is here that the problem lies.

Decomposition of the Lot-Size Problem

The decomposition method of Dantzig and Wolfe applies directly to linear programs in which the set of constraints can be partitioned into $(p + 1)$ subsets where p of these subsets are mutually independent. That is to say, each subset involves a different set of variables. The solution to a given problem requires replacing the problem with $(p + 1)$ smaller linear programming problems, i.e., with p independent problems and one connection problem.

The equations (4.0)–(4.3) can be rewritten as:

$$\text{minimize} \quad C\theta,$$

subject to:

$$A\theta = D, \qquad \theta \geqq 0$$

We can now divide A into two sub-matrices, so that $A\theta = D$ is replaced by:

$$L\theta = d_0, \tag{5.0}$$

$$A_1\theta = d_1. \tag{5.1}$$

Where L has KT rows corresponding to the rows in (4.1) and can be partitioned vertically into I sub-matrices L_i $(i = 1, \cdots, I)$. A_1 has the I rows corresponding to (4.2). A_1 can also be partitional into sub-matrices A_{i1} $(i = 1, \cdots I)$, each consisting of a single row. This situation is shown in Figure 1.

We now transform the problem into one with $KT + 1$ rows and a great many columns by the following reasoning, due to Dantzig and Wolfe [4]. The only θ vectors we need to consider as solutions to (5.0) are those that solve (5.1); clearly the solution space to (5.1) is bounded. Thus we can, in principle, give

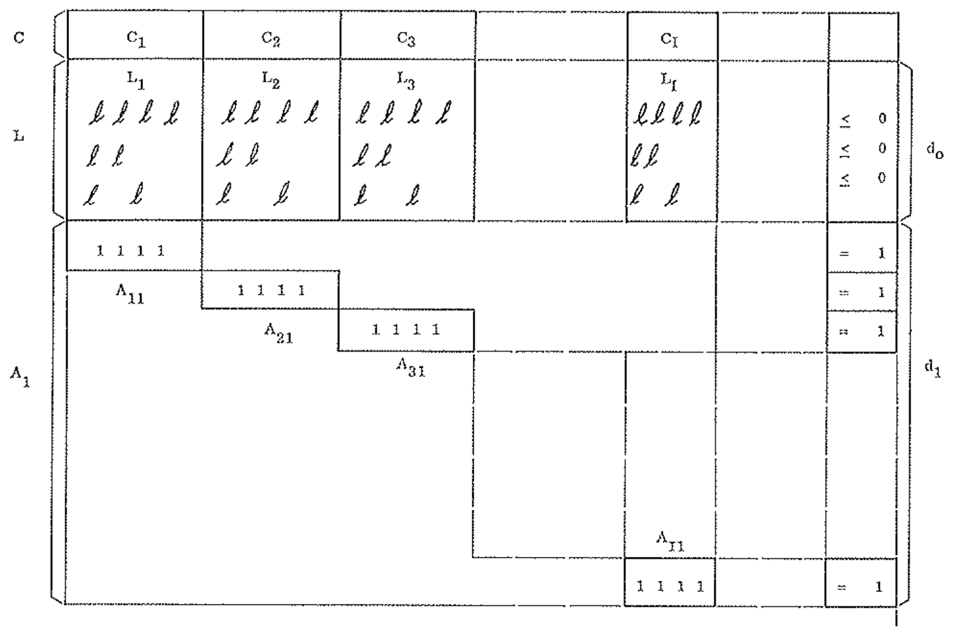

FIG. 1. Tableau description of problem (5.0)–(5.1)

a complete list of vertex solutions ϕ^q to (5.1) and any solution, whatsoever, to (5.1) is a weighted combination of these ϕ^q with non-negative weights λ_q totaling 1; so any feasible solution is of the form:

$$\phi = \textstyle\sum_q \lambda_q \phi^q, \qquad \lambda_q \geqq 0, \ \sum_q \lambda_q = 1, \tag{6.0}$$

By substitution in (4.0)–(4.3), the original problem can be written as:

$$\text{Min } z = \textstyle\sum_q \bar{C}_q \lambda_q , \tag{7.0}$$

subject to:

$$\textstyle\sum_q \lambda_q L_q = d_0 , \tag{7.1}$$

$$\textstyle\sum_q \lambda_q = 1, \tag{7.2}$$

$$\lambda_q \geqq 0, \tag{7.3}$$

Where $\bar{C}_q = C\phi^q$ and $L_q = L\phi^q$.

The relations in (7.0)–(7.3), define a new linear programming problem. In passing from (4.0)–(4.3) to (7.0)–(7.3), the number of equations is reduced from $I + KT$ to $KT + 1$ at the cost of adding a large number of variables. There is now one for every vertex solution of (5.1).

We now want to solve the problem of (7.0)–(7.3) by employing the simplex algorithm. Suppose we have a basic feasible solution, which is denoted as B, and we wish to perform a simplex iteration. We use the revised form of the simplex; the top row of the inverse of the current basis provides the prices

$$\mu = (\pi_1 , \cdots , \pi_{KT+1}) = (\bar{\pi}, \pi_{KT+1})$$

Now the criterion (cf. Ref. (2), (11), (15)) for introducing a column vector into the current basis matrix B is determined from the scalar products of μ with the column vectors $(L_q, 1)$. In fact the column selected is the column which maximizes $-\tilde{C}_q + \mu \cdot (L_q, 1)$. Thus the problem of selecting a column is

$$\begin{aligned}\text{Max}_q -\tilde{C}_q + \mu \cdot (L_q, 1) &= \text{Max}_q (-C\phi^q + \mu \cdot (L\phi^q, 1)) \\ &= \text{Max}_q (-C + \bar{\pi}L) \cdot \phi^q + \pi_{KT+1}\end{aligned}$$

Thus we are maximizing a linear function over all vertices, ϕ^q or equivalently over a convex body. Therefore, this problem is again a linear programming problem of the following form:

$$\text{Max } (-C + \bar{\pi}L) \cdot \phi + \pi_{KT+1}. \tag{8.0}$$

subject to

$$A_1\phi = d_1, \qquad \phi \geqq 0 \tag{8.1}$$

This is a large linear programming problem; however, we see that this can be solved easily if we split L into submatrices corresponding to each i^{th} item. If we let L_{ij} be a column of L corresponding to the component θ_{ij}, of θ, (8.0), (8.1) splits into a series of subproblems each of the form:

$$\text{Max } (-C_{i1} + \bar{\pi} \cdot L_{i1}, -C_{i2} + \bar{\pi} \cdot L_{i2}, \cdots, -C_{in_i} + \bar{\pi}L_{in_i}) \cdot (\theta_{i1}, \theta_{i2}, \cdots, \theta_{in_i}) \tag{9.0}$$

subject to

$$\textstyle\sum_{j=1}^{n_i} \theta_{ij} = 1 \qquad \text{(all } i\text{)} \tag{9.1}$$

The solution is simply to set $\theta_{ij} = 1$ for the j value for which $-C_{ij} + \bar{\pi} \cdot L_{ij}$ is maximal, and set all other $\theta_{ij} = 0$. Thus the problem reduces to that of finding, from all possible feasible production schedules for the i^{th} item the one for which $-C_{ij} + \bar{\pi} \cdot L_{ij}$ is maximal. This is then repeated for each i. It is at this point that we run into the difficulties caused by the multitude of possible production schedules.

Generation of Alternative Labor Input Vectors

The procedure for solving (9.0)–(9.1) above is essentially a column-selection procedure performed for each item. However, when the number of periods for which production must be planned is $T = 10$, the maximum number of possible production schedules is $2^{T-1} = 512$. If production is to be planned for 500 items, 256,000 columns must be generated beforehand; and in doing the column selection, 256,000 scalar products, $C_{ij} + \bar{\pi} \cdot L_{ij}$, must be computed. Even with the help of a large-scale digital computer, this calculation will tend to get out of hand.

Fortunately this maximization problem, finding *max* $-C_{ij} + \bar{\pi} \cdot L_{ij}$ can be handled by special methods. The problem for each i is to find the "dominant" schedule j that minimizes

$$C_{ij} + (-\bar{\pi}\cdot L_{ij}) = \sum_{k\tau} - \bar{\pi}_{k\tau}(a_{ik}\delta(x) + b_{ik}x_{ij\tau}) + \sum_\tau c_{i\tau}x_{ij\tau} = \sum_\tau \{A_{i\tau}\delta(x) + B_{i\tau}x_{ij\tau}\} \tag{10}$$

where $\delta(x) = 0$ if $x = 0$ and is 1 otherwise, and

$$A_{i\tau} = \sum_k - \bar{\pi}_{k\tau}a_{ik}$$

$$B_{i\tau} = (\sum_k - \bar{\pi}_{k\tau}b_{ik}) + c_{i\tau}$$

since the $\pi_{k\tau}$ are $\leqq 0$, it follows that $A_{i\tau} \geqq 0$, $B_{i\tau} \geqq 0$. The problem of minimizing (10) over production levels that satisfy the demands $r_{i\tau}$ in each time period is precisely the dynamic economic lot size model studied by several authors [1], [12], [16], [17]. We follow the approach of Wagner and Whitin [17].

In viewing (10) as an economic lot size problem $A_{i\tau}$ becomes the setup cost and $B_{i\tau}$ the marginal cost of production. To minimize (10) we introduce the function $C_{i\tau}(y)$. $C_{i\tau}(y)$ is the minimum cost of filling all requirements $r_{i\tau}$ up to and including the τ^{th} period and having on hand at the end of the τ^{th} period an amount y of extra production. $C_{i\tau}(y)$ can be obtained recursively from

$$C_{i1}(y) = A_{i1}\delta(y + r_{i1}) + B_{i1}(y + r_{i1}) \tag{11a}$$

and

$$C_{i\tau}(y) = \min\{A_{i\tau}\delta(z) + B_{i\tau}(z) + C_{i\tau-1}(y + r_{i\tau} - z)\}, \quad 0 \leqq z \leqq y + r_{i\tau}, \qquad \tau > 1 \tag{11b}$$

The calculation in (11b) is greatly reduced by two observations whose justification can be obtained from [17].

(i) It is only necessary to consider the two z values $z = 0$ and $z = y + r_{i\tau}$ when performing the minimization.

(ii) $C_{i\tau}(y)$ is needed only for the $T - \tau + 1$ y values

$$V_{ip} = \sum_{s=\tau}^{s=p} r_{i,s}$$

where $\tau \leqq p \leqq T$.

Thus to obtain $C_{iT}(0)$ which is the minimum cost of filling all orders it is only necessary to compare values corresponding to $z = 0$ and $z = y + r_{i\tau}$ approximately $T(T + 1)/2$ times. Backtracking to obtain the $x_{ij\tau}$ that gave that cost is only very small additional calculation. This then is the amount of calculation that is substituted for evaluating the 2^{T-1} scalar products.[1]

We have just explained how to solve the individual problems (9.0)–(9.1). When solved for all i these provide all the components θ_{ij} for the vector ϕ^q satisfying (8.0)–(8.1). Then $L_q\phi^q$ is the new vector to be introduced into the basis.

At this time, we summarize the work as follows: by a Dantzig and Wolfe decomposition followed by an application of the dynamic programming technique just discussed, a problem originally calling for simplex operations on a $(I + KT) \times (I + KT)$ matrix, and the investigation of scalar products with

[1] For computational experience with this recursive calculation, see [1].

columns as numerous as production schedules, is reduced so simplex operations on a $(KT + 1) \times (KT + 1)$ matrix, and a succession of I associated dynamic programming calculations.

Next, let us turn to the question of integer θ_{ij}. Fortunately, as Manne [14] has observed this problem tends to take care of itself. For in the system, there are $I + KT$ equations and, therefore at most, $I + KT$ positive variables in the optimal solution. There must be at least one positive variable appearing with a non-zero coefficient in each of the equations (4.1). This accounts for I of the $I + KT$ positive variables, so there are at most KT values of i for which more than one θ_{ij} is positive. If I is much larger than KT, this means that almost always only one θ_{ij} is positive for a fixed i. Therefore, it must be an integer with value 1. Fortunately, then, there are at most KT variables at proper fractional values. These may be treated by some arbitrary rounding process.

An Experimental Computer Code

An experimental computer code was developed for overcoming the computational problem of the large number of constraint rows and activity vectors. We are mainly concerned with accomplishing the task by the techniques of Dantzig and Wolfe, and by the dynamic programming technique for creating new least-cost labor hour coefficient vectors.

In testing the computer program, the economic lot size and work-force planning model studied by Dzielinski, Baker and Manne [5] was used. That is, the problem not only includes (4.0)–(4.3), but also variables that allow the work force to vary. Before writing out in full the equations on which the actual calculations were made we will review some notations. It should also be borne in mind that these equations correspond to (7.0)–(7.3), i.e., the master coordinating program of the Dantzig and Wolfe decomposition and not to the original linear programming formulation (4.0)–(4.3).

We will refer to "Production Plan Vectors". These are vectors $L\phi^q$ of our previous discussion. Here they are written as vectors $P_{\bar{q}}$ with elements $p_{k\tau\bar{q}}$; $p_{k\tau\bar{q}}$ is the labor hours of type k required in period τ under this plan, and is $= \sum_i l_{ij(i)k\tau}$, where $j_{(i)}$ is the index of the production schedule selected for item i by solving (9.0)– – (9.1) $P_{\bar{q}}$ also contains a 1 in the $KT + 1$ position and a cost $C_{\bar{q}} = \sum_i C_{ij(i)}$. With each $P_{\bar{q}}$ we associate the variable $\lambda_{\bar{q}}$, satisfying $\lambda_{\bar{q}} \geqq 0$ and $\sum_{\bar{q}} \lambda_{\bar{q}} = 1$.

Now we can write the full equations used which are:
Minimize

$$\sum_{\bar{q}} \lambda_{\bar{q}} C_{\bar{q}} + \sum_{k,\tau} [\textstyle\sum_r R^r_{k\tau} W^r_{k\tau}] + \sum_{k,\tau} [\Gamma^+_{k\tau} W^+_{k\tau} + \Gamma^-_{k\tau} W^-_{k\tau}] \tag{12.0}$$

Subject to

$$\sum_{\bar{q}} \lambda_{\bar{q}} p_{k\tau\bar{q}} - \sum_r H_k{}^r W^r_{k,\tau} \leqq 0 \qquad k = 1, \cdots, K, \tau = 1, \cdots, T \tag{12.1}$$

$$\sum_{\bar{q}} \lambda_{\bar{q}} p_{s\bar{q}} = \sum_{\bar{q}} \lambda_{\bar{q}} = 1 \qquad s = KT + 1 \tag{12.2}$$

$$\sum_r W^r_{k\tau} - W^+_{k\tau} + W^-_{k\tau} = \sum_r W^r_{k\tau-1} \qquad (\text{all } k, \tau), \tag{12.3}$$

	PRODUCTION PLAN VECTORS	WORK FORCE VARIABLES			CHANGE IN WORK FORCE	RIGHT HAND SIDE
		FIRST SHIFT	SECOND SHIFT	THIRD SHIFT		
	λ λ … λ λ	w^1 w^2 w^1 .. w^1 w^2	w^3 w^4 . w^3 w^4	w^5 w^6 . w^5 w^6	w^+ w^- w^-	
COST ROW	$\sum_i c$ $\sum_i c$ … $\sum_i c$ $\sum_i c$	R^1 R^2 R^1 .. R^1 R^2	R^3 R^4 . R^3 R^4	R^5 R^6 . R^5 R^6	r^+ r^- $\cdot r^-$	
τ k 1 1	$\sum_i \ell$ $\sum_i \ell$ … $\sum_i \ell$ $\sum_i \ell$	-H -H . .	-H -H . .	-H -H . .	. .	$\leqq 0$
2 1	$\sum_i \ell$ $\sum_i \ell$ … $\sum_i \ell$ $\sum_i \ell$	-H . .	. .	. .	. .	$\leqq 0$
. .	. .Total Labor. .. . .Requirements ..	. .Labor . . . . .Availability	. Labor . . Availability	. Labor . . . Availability.	....	. .
. .	....	....	....	....	.....	.
T K	$\sum_i \ell$ $\sum_i \ell$ ⋯ $\sum_i \ell$ $\sum_i \ell$	. -H -H .	-H -H	-H -H		$\leqq 0$
CONVEX CON-STRAINT	1 1 … 1 1	.	.	.	.	=1
	…	1 1 .	1 1 .	1 1 .	-1 1 .	$= \sum_r w^r_{k,\tau-1}$
	…	-1 -1 1 .	-1 -1 .	-1 1 .	.	= 0
....	....	. Labor Balance	. Labor Balance	. Labor Balance	.....	...
	…	. 1	. 1	. 1	.	= 0
		. -1 1	. -1 1	. -1 1	. 1	= 0
	…	1 1 .	.	.	.	$\leqq M_1$
....	....	. Labor Capacity	....	....	.	.....
	…	. 1 1	.	.	...	$\leqq M_k$
	…	.	1 1 .	.	.	$\leqq M_1$
....	....	....	Labor Capacity	....	...	.....
	…	.	. 1 1	.	.	$\leqq M_k$
	…	.	.	1 1 .	.	$\leqq M_k$
....	....	....	....	Labor Capacity	....	.....
	…	.	.	. 1 1	.	$\leqq M_k$

FIG. 2. Tableau description of problem (12.0)–(12.4)

$$
(12.4) \qquad \begin{aligned}
&W^1_{k\tau} + W^2_{k\tau} \leqq M_k, && \text{(First Shift)}\\
&W^3_{k\tau} + W^4_{k\tau} \leqq M_k, && \text{(Second Shift)} \quad (\text{all } k,\ \tau)\\
&W^5_{k\tau} + W^6_{k\tau} \leqq M_k, && \text{(Third Shift)}\\
&\lambda_{\bar{q}},\ W^r_{k\tau},\ W^+_{k\tau},\ W^-_{k\tau}, && \text{all} \geqq 0.
\end{aligned}
$$

Figure 2 depicts the model of (12.0)–(12.4) in tableau form.

In the above $W^r_{k\tau}$ is the number of workers of Type k and payment class r working in period τ.

The superscript $r = 1$ means straight time for the first shift, $r = 2$ means straight time and overtime on the first shift. The superscripts 3, 4, 5, 6 refer to the same payment classes on the second and third shifts. H_k^r is the number of hours this type of worker will work in a period and $R^r_{k\tau}$ is the corresponding labor rate cost coefficient.

TABLE 1

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
I: Parameters that Determine Size of Problems										
(1) I: Number of Items	35	963	428	428	428	428	100	100	100	100
(2) K: Number of Facilities	2	2	2	2	2	2	1	1	1	1
(3) T: Number of Periods	3	3	3	5	7	8	12	12	12	12
(4) S: Number of Shifts	2	2	2	2	2	2	2	2	2	2
(5) $(2 \cdot K \cdot T) \cdot (S+1)$: Work Force Columns	36	36	36	60	84	96	72	72	72	72
(6) $(K \cdot T) \cdot 3$: Slack Columns	18	18	18	30	42	48	36	36	36	36
(7) $(S+2) \cdot (K \cdot T)+2$ Number Interacting Constraint Rows	26	26	26	42	58	66	50	50	50	50
(8) $(2^{T-1}) \cdot I$: Item Production Schedules	140	3847	1712	6784	27392	54434	204800	204800	204800	204800
II: Size of Problems as Ordinary Linear Programming Problems										
(9) (1)+(7): Number of Rows	61	989	454	466	486	494	150	150	150	150
(10) (5)+(6)+(8): Number of Columns	194	3901	1766	6874	27518	54928	204908	204908	204908	204908
III: Size of Problems with Decomposition and Dynamic Programming										
(11) (7): Number of Rows	27	27	27	43	59	66	51	51	51	51
(12) (5)+(6): Number of Columns	54	54	54	90	126	144	108	108	108	108
(13) (1) $\times$ $(T(T+1)/2)$: Number of Elementary Steps in DP Alg.	210	5778	2568	6420	11984	15408	7800	7800	7800	7800

IV: Computations of Problems with Decomposition and Dynamic Programming

(14) Total Time to Compute Optimal Solution (Minutes of 7090 time)......	3.36*†	24.5*†	5.66*	9.0*	10.70*	17.92*	5.90*	8.87*	8.07*	9.00*
(15) Total Phase II Iterations.................	48	34	22	35	52	65	45	49	46	51
(16) Average time to create a Production Plan (Minutes).................	.07†	.72†	0.264	0.345	0.483	0.510	0.183	0.225	0.223	0.215
(17) Number of Production Plans Created.........	16	12	19	24	20	29	26	34	31	36

* This does not include 7090 time to setup the problem and output the results.

† This problem was run using a version of the dynamic programming subroutine that was later improved. The improved routine, used in the remaining problems, requires about ⅔ less time than the original routine.

problem having a very large number of columns, the usual rules of thumb for simplex computations often do not apply, the computation may have a very long "tail", i.e., have a tremendous number of almost optimal solutions before coming out to the optimum, see for example [9]. Although attaining the actual optimum is not a vital practical concern in most cases, it is important in this problem as we explain below in our discussion of split lots.

Fortunately the computational results were in fact encouraging, see Table 1. All ten of the test problems terminated and in reasonable runs. It is quite clear, from looking at Part II of Table 1 that many of these problems could not have been run as standard linear programming problems. In fact, both the decomposition and the use of dynamic programming were necessary.

We now explain the importance of actual termination in our problem. The number of split lots, i.e., items for which there are more than one production schedule, can be bounded in the complete work force model just as in the simplex model. The number of split lots in a basic feasible solution to the direct LP formulation would be at most $(S + 2)KT$, where S is the number of shifts. Now a basic feasible solution obtained from (12.0)–(12.4) will not usually be a basic solution in the direct LP formulation. Consequently, a virtually unlimited number of split lots could occur making the solution useless. Fortunately, as long as the problems have unique optima, the optimal solution to (12.0)–(12.4) will give the same schedules as the optimal solution to the original LP, and since this latter is basic, the optimal solution to (12.0)–(12.4) also has at most $(S + 2)KT$ split lots.

This limit would not apply if we had been obliged to stop at a non-optimal solution.

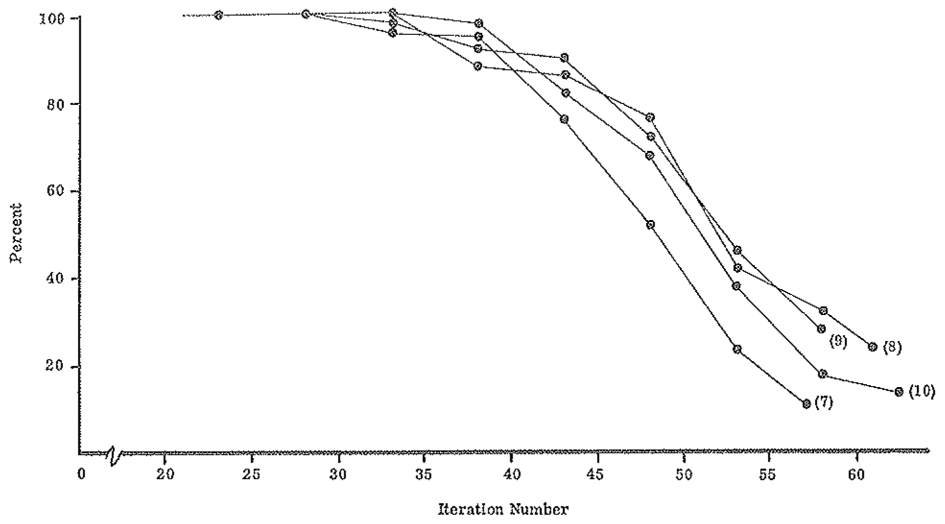

FIG. 4. The percentage of items with split lots in the solution to problems (7), (8), (9), (10).

In Figure 4 we give a graph showing the evolution of the number of split lots in the solution of four problems. In these problems, even before the optimum was reached, the number of split lots had become quite small. This raises the possibility that even if problems were encountered which did not terminate in a reasonable number of steps the near-optimal solutions could still be used.

Appendix A

The Lot-Size Programming program is outlined in schematic form in Figure 3. It is a combination of several FORTRAN and FAP programming language routines. The entire system compiles and executes under the FMS-II job processor on the IBM 7090/94 Data Processing System.

It includes:

1. PARTC—a program that inputs from cards, problem parameters, prepared column vector data on the work force variables, and right hand side elements.
2. PARTD—a program that generates the artificial variables cost row, slack and artificial vectors, and the initial solution.
3. PARTE—a program that generates a column row dictionary for the slack and artificial vectors, and writes this table on the problem output tape.
4. PARTG—this program is the main computation and linkage control program of the linear programming algorithm. It controls the subprograms that generate the least cost item production schedule and the production plan vector on any given iteration, it performs certain arithmetic operations, and controls the subprograms which output the values for the work force variables, production plan vectors, and item production schedules in the solution to a given problem.

The following programs are subprograms of PARTG and their functions are as follows:

PARTH—a program that controls the subroutine DYMP. It also creates the production plan vector and stores it in the matrix data area for the master LP problem for consideration by the simplex algorithm on any given iteration.

DYMP—a subprogram of PARTH that generates the individual least cost item production schedule by a dynamic programming algorithm, computes the associated labor hours input coefficients and cost coefficients for the given schedule, and stores it for PARTH. This is the program that uses the dual variables, item demands, set-up times, unit process times, and unit material costs as input to generating the least cost schedules.

PARTI—an output program concerned with giving the values of the work force variables, change in work force variables and slack variables in the current solution on any given iteration or the final optimal solution of the problem.

PARTK—a program that writes out the values and identifies the production plan vectors in the current solution on any given iteration or in the optimal solution.

PARTJ—a program that determines which item production schedules make up the production plan vectors in the current solution on any given iteration or in the optimal solution, and writes out the corresponding production quantities for each time period.

Explicit details and operating instructions for using the program will be part of a forthcoming publication.

References

1. Bhatia, A. and Garg, A., "Application of Dynamic Programming to a Class of Problems in Inventory Control," *Journal of Industrial Engineering*, 11, No. 6, (1960), pp. 500–512.
2. Charnes, A. and Cooper, W. W., *Management Models and Industrial Applications of Linear Programming, Vol. I and Vol. II*, John Wiley & Sons, New York.
3. Dantzig, G. B. and Wolfe, P., "Decomposition Principle for Linear Programs," *Operations Research*, Vol. 8, (1960), pp. 101–111.
4. ———, "A Machine Job Scheduling Model," *Management Science*, (1961), pp. 191–196.
5. Dzielinski, B. P., Baker, C. T. and Manne, A. S., "Simulation Tests of Lot Size Programming," *Management Science*, Vol. 9, No. 2, (1963), pp. 229–258.
6. Ford, L. R. and Fulkerson, D. R., "A Suggested Computation for Maximal Multicommodity Network Flows," *Management Science*, Vol. 5, No. 1, (1958), pp. 97–101.
7. Gass, S., *Linear Programming*, McGraw-Hill, New York, 1958.
8. Gilmore, P. C. and Gomory, R. E., "A Linear Programming Approach to the Cutting Stock Problem," *Operations Research*, Vol. 9, No. 6, (1961), pp. 849–859.
9. ———and———, "A Linear Programming Approach to the Cutting Stock Problem—Part II," *Operations Research*, Vol. 11, No. 6, (1963), pp. 863–888.
10. Gomory, R. E., "Large and Non-Convex Problems in Linear Programming," *Proceedings of the Fifteenth Symposium of the American Mathematical Society*, American Mathematical Society, 193 Hope Street, Providence, Rhode Island, (1963), pp. 125–139.
11. Hadley, G. M., *Linear Programming*, Addison-Wesley, Reading, Massachusetts, 1962.
12. Kerner, H., "Scheduling for Known but Irregular Batch-wise Demand," *Operations Research Quarterly*, Vol. 12, No. 4, (1961).
13. Levitan, R. E., "A Note on Professor Manne's Dominance Theorem," *Management Science*, Vol. 4, (1958), pp. 115–135.
14. Manne, A. S., "Programming of Economic Lot Sizes," *Management Science*, Vol. 4, (1958), pp. 115–135.
15. Simonnard, M., *Programmation Lineaire*, Dunod, Paris, 1962.
16. Wagner, D. E. and Shetty, E. M., "Solution to a Production Scheduling Problem with Fixed Costs," *Operations Research Quarterly*, Vol. 13, No. 1, (1962).
17. Wagner, H. M. and Whitin, T. M., "A Dynamic Version of the Economic Lot Size Model," *Management Science*, Vol. 5, (1958), pp. 89–96.