Discussed is an optimal programming model for lot-size, inventory, and work-force planning over a finite planning horizon for assembly-type production. The object is to prepare a minimum cost lot-size and work-force plan that meets the deterministic demands within the resource constraints. Planning for end-items, components, and overtime is included.

The main feature of the model is the ability to plan for assembly production having precedences with nonlinear (set-up) costs using essentially linear programming computations.

Programming for economic lot-sizes with precedences between items

by S. Gorenstein

In the manufacture of certain products such as computers, turbines, and automobiles, various components and subassemblies must be combined in a specified assembly order over a period of time. Certain items must be on hand before others can be produced, and these requirements are determined by the product structure. Management's objective is to turn out timely products in the most economic way possible, and they need to determine production quantities known as lot-sizes to achieve this objective. For these types of products, a multi-item lot-size determination is necessary.

In this paper, we are concerned with an assembly model whose purpose is to provide minimum cost lot-size, inventory, and work-force plans for a particular type of production environment. The model is a linear program for determining multi-item economic lot-sizes to meet deterministic demands over a finite planning horizon with precedence or order relations between the items to be produced, including set-up costs.

The application of linear programming to economic lot-size decisions for the production of independent items made by Manne[1] was extended by Dzielinski, Baker, and Manne[2] and

Dzielinski and Gomory[3] to include inventory and work-force decisions. In this work, we further extend the model to cover nonindependent commodities, that is, order relations or precedences exist in that certain items may have to be on hand before other items can be made, as in production for assemblies. We describe a program augmented by the precedence or order relations.

The object of the program is to determine the economic lot-sizes and the work force required to meet given demands for items over a planning horizon of T time periods. Considering material, labor (unit and setup), hiring and firing costs, it determines the lot-size for each item to be produced in each time period of the planning horizon.

In this paper, we show in detail how the constraints generated by the order relations between items in the program are constructed. We then discuss solution methods for the program and show explicitly how the Dantzig and Wolfe[4] decomposition principle can be applied to the solution. We also show how the decomposition subproblems can be used to generate the production vector to enter the basis of the program with the order relations, without having explicitly available the columns of the original linear program. Then we consider the question of integer solutions to the program.

The constraints generated by the order relations among items are not specifically included in the constraint matrix, but the generation of item requirements takes them into account. This is done by an "explosion" within the linear program from the solutions of the subproblems for higher-level items to get the demands for the lower-level items. This maintains the order relations and enables us to get the same bound on the number of noninteger solutions as that obtained by Grigoriadis[5] for the program without order relations.

Description of the program

Following along the same lines as that in Manne,[1] the activities of the program are the dominant production vectors expressed in terms of labor-hour requirements. There are various payment classes of labor: straight-time workers, straight time plus overtime workers, and workers on different shifts, and workers can be hired and fired. The resource constraint is represented by the total number of workers that can work at a facility, which is assumed to also represent the capacity of the facility in terms of its equipment. Thus, it is immaterial whether labor hours or machine hours are considered as a resource. We use labor hours since we are incorporating work-force planning into the program.

The linear program requires the following constraint equations:

- Constraints to meet the labor-hour requirements of the dominant production schedules (represented by Equation 2)
- Constraints to preserve the order relations among the items (represented by Equation 3)
- Constraints to select production schedules from among the dominant ones (represented by Equation 4)
- Labor balance equations which ensure that the number of workers at the beginning of a period equals the number at the end of the previous period (represented by Equation 5)
- Capacity constraints (represented by Equation 6)

The variables in the program select production schedules for each item and determine the size of the work force needed of each payment class in each time period – straight time, overtime, shift. Also, hiring and firing decisions are indicated. The costs include the material costs associated with the production schedules, the various labor costs, and the hiring and firing costs.

In summation, we can say that the linear program is a production planning model that incorporates lot-size, inventory, and workforce decisions to minimize overall costs.

statement of the program

Except for Equation 3, the following program is that given by Dzielinski and Gomory.[3]

Minimize

$$\sum_i \sum_j C_{ij}\,\theta_{ij} + \sum_k \sum_\tau \sum_r R^r_{k\tau}\,W^r_{k\tau} + \sum_k \sum_\tau (\Gamma^+_{k\tau}\,W^+_{k\tau} + \Gamma^-_{k\tau}\,W^-_{k\tau}) \qquad (1)$$

where θ_{ij} represents the fraction of the requirement for the ith item supplied by the jth production schedule for that item, $W^r_{k\tau}$ is the number of workers of payment class r to be employed at facility k in period τ, $W^+_{k\tau}$ is the number of additional workers to be hired, and $W^-_{k\tau}$ is the number of workers to be fired. The constant C_{ij} is the material and holding cost associated with schedule j for item i, $R^r_{k\tau}$ is the cost of a worker of payment class r at facility k during period τ, $\Gamma^+_{k\tau}$ is the cost of hiring a worker, and $\Gamma^-_{k\tau}$ is the cost of firing a worker.

Minimization of the objective function is subject to

$$\sum_i \sum_j \ell_{ijk\tau}\,\theta_{ij} - \sum_r H^r_{k\tau}\,W^r_{k\tau} \leq 0, \qquad \begin{matrix} k = 1, 2, \cdots, K \\ \tau = 1, 2, \cdots, T \end{matrix}$$

where $\ell_{ijk\tau}$ is the number of labor hours required at facility k in period τ to produce item i according to schedule j, and $H^r_{k\tau}$ is the number of labor hours provided by a worker.

$$\sum_i \sum_j y_{ijm}\,\theta_{ij} \leq 0, \qquad m = 1, 2, \cdots, P \qquad (3)$$

where y_{ijm} represents the coefficients generated by the order relations to ensure technological feasibility.

$$\sum_{j=1}^{J_i} \theta_{ij} = 1, \qquad i = 1, 2, \cdots, I \tag{4}$$

$$\sum_{r} W_{k\tau}^{r} - W_{k\tau}^{+} + W_{k\tau}^{-} - \sum_{r} W_{k,\tau-1}^{r} = 0, \qquad \text{all } k, \tau \tag{5}$$

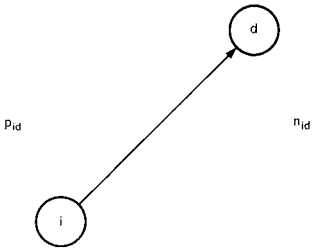
$$\begin{aligned} &\sum_{r=1}^{p} W_{k\tau}^{r} \leq M_{k\tau}^{1}, && \text{first shift} \\ &\sum_{r=p+1}^{q} W_{k\tau}^{r} \leq M_{k\tau}^{2}, && \text{second shift} \\ &\sum_{r=q+1}^{v} W_{k\tau}^{r} \leq M_{k\tau}^{3} && \text{third shift} \end{aligned} \tag{6}$$

where $M_{k\tau}^{s}$ is the bound on the total number of workers of all payment classes on shift s.

$$\theta_{ij}, W_{k\tau}^{r}, W_{k\tau}^{+}, \mathrm{W}_{k\tau}^{-} \geq 0, \qquad \text{all } k, \tau$$

We now discuss Equation 3, the one generated by the order relations between items. The order relations are technological in that we are concerned with the manufacture of a product which is assembled from other manufactured and/or purchased items, called details or subassemblies. It also happens that there are different levels of subassemblies; that is, some subassemblies are manufactured from other subassemblies that are on a lower level in the assembly process. In order to manufacture the final assembly or end product, known quantities of subassemblies are required and these must be on hand a known number of time periods in advance of the time when the assembly in which they are used is to be completed. The entire technological structure is known, and a given demand over a number of time periods for end products will generate a demand over time for subassemblies.

Figure 1 Product structure



However, we have the additional nontrivial complication that the order relationships must be maintained. For example, if the demand for end product A in period 10 is to be met by production in period 9, then all the subassemblies required for item A must be produced in time so that they will be available for the production of A in time period 9. Thus, producing A earlier requires the earlier production of its components. The object is to produce according to a schedule that will minimize the costs of meeting the (deterministic) demands over a number of time periods for end products, subject to the capacity constraints and the order relations.

We now construct Equation 3 to ensure that the order relations will be maintained. Consider a portion of the product structure as shown in Figure 1. This indicates that a quantity n_{id} of item i

is needed p_{id} time periods in advance to make or assemble item d.

Suppose further that there were:

J_d dominant production schedules for product d
J_i dominant production schedules for product i

Then we would have the constraint for the first period

$$\sum_{j=1}^{J_i} \theta_{ij}\, x_{ij1} \geq n_{id} \sum_{j=1}^{J_d} \theta_{dj}\, x_{djp_{id}+1}$$

since the left side is the quantity of item i produced in period 1, as specified by the solution to the program, and the right side is the number of item i required in period 1 to produce the programmed amount of item d in period $p_{id} + 1$.

Thus, these constraints can be stated as

$$\sum_{j=1}^{J_i} \theta_{ij} \sum_{\tau=1}^{m} x_{ij\tau} \geq n_{id} \sum_{j=1}^{J_d} \theta_{dj} \sum_{\tau=1}^{m} x_{djp_{id}+\tau} \qquad (7)$$

$m = 1, 2, \cdots, T - p_{id}$

which gives $T - p_{id}$ equations, where $x_{ij\tau}$ is the quantity of item i to be produced in period τ according to schedule j.

Further, if n_{id} of product i are required for product d at p_{id} periods in advance and n_{ie} of product i are required for product e at p_{ie} periods in advance, the restrictions become

$$\sum_{j=1}^{J_i} \theta_{ij} \sum_{\tau=1}^{m} x_{ij\tau} \geq n_{id} \sum_{j=1}^{J_d} \theta_{dj} \sum_{\tau=1}^{m} x_{djp_{id}+\tau} + n_{ie} \sum_{j=1}^{J_e} \theta_{ej} \sum_{\tau=1}^{m} x_{ejp_{ie}+\tau} \qquad (8)$$

for $m = 1, 2, \cdots, \max(T - p_{id}, T - p_{ie})$

Thus, wherever a part is used in other parts, a set of restrictions such as these would have to be met.

Solution methods

general considerations

The method adopted for the solution of the program in the expressions 1 through 6 depends on a number of factors. In the rare case where the number of variables is small, say under 100, a direct integer programming approach might be considered. Several integer programming algorithms—cutting plane, branch and bound, partial enumeration—are available, but there have been no demonstrated successes with large problems. While the simplex algorithm has performed very well in practice for all linear programs, that is, the algorithm has reached an optimality indication within a reasonable amount of time, the same cannot be said for the integer programming algorithms when it comes to large problems.

However, where direct integer programming is not practicable, as is true in most cases of interest, we can seek a linear programming solution and then apply some heuristic methods to resolve the problem of getting integer solutions. But we shall see later that in most cases of interest, and specifically those for which this model is designed, where I, the number of items, is much larger than $2KT$, the problem of integer solutions mostly takes care of itself.

In the case of the program without the precedence requirements (Equation 3), that is, the program represented by Equations 1, 2, 4, 5, and 6, it can be shown that there will be at most $2KT$ noninteger solutions for the I production schedules. This follows from there being $(3 + 2)KT + I = E$ equations in the system and therefore, at most, that many vectors in the basis, or that many positive variables. However, the I constraints of Equation 4 require at least I positive variables, and the $3KT$ constraints of Equation 6 require at least $3KT$ positive variables. So there can be at most $E - (I + 3KT) = 2KT$ cases where more than one variable enters at a positive level to satisfy the constraints of Equations 4 and 6, and these can be any of the variables: production schedule or work force. But from the nature of the constraints of 4, this means that there will be integer solutions for production schedules except for, at most, $2KT$ cases, where there is a weighting of the production vectors for an item. When I is much larger than $2KT$, some simple rounding process can be used to select production schedules where there are such noninteger solutions. Something as simple as using that particular convex (weighted) combination of the schedules should hardly affect the feasibility or optimality of the solution.

But where we have precedence requirements the situation may not be so simple. We have now $E + P$ equations, where P is the number of precedence requirements that we have to consider. Of course, there is a theoretical bound, $I(T - 1)$, on the number of such equations, but this is not very helpful since it is comparatively large. However, in particular applications it may be that there are *relatively* (relative to I) few such constraints since not all details or subassemblies are required for higher assemblies but may go directly into the end product. Therefore, it may be worth our while to determine the magnitude of P and see how it compares to I. If we still find that I is much greater than $2KT + P$, which is now the maximum number of noninteger solutions for production schedules, we can still use some simple rounding process on the solution of the linear program. Further, even if this condition does not hold, $2KT + P$ is an *upper* bound on the number of noninteger solutions; it does not mean that there will actually be that many. It may pay to solve it as a linear program and see how many noninteger solutions there actually are, and then determine if some rounding process will

work adequately or if other methods are required. Such other methods are discussed later and in Gorenstein.[6]

decomposition

While it is possible to view the program in expressions 1 through 6 as a straightforward linear program, an application of the Dantzig and Wolfe[4] decomposition principle would provide computational advantages in most cases when there is a large number of items.

We rewrite the problem in matrix form as

$$\text{Min } C\theta + RW + \Gamma F \tag{9}$$

$$A_1\theta + A_2W + A_3F = b \tag{10}$$

$$B_1\theta = b_1 \tag{11}$$

$$B_2W = b_2 \tag{12}$$

$$\theta, W, F \geq 0$$

Feasible points can be written as convex combinations of extreme points

$$\theta = x_1 = \sum_k \lambda_k x_{1k}, \sum_k \lambda_k = 1, \qquad \lambda_k \geq 0 \tag{13}$$

$$W = x_2 = \sum_k \mu_k x_{2k}, \sum_k \mu_k = 1, \qquad \mu_k \geq 0 \tag{14}$$

Then, substituting this in Equations 9 through 12, and setting

$$P_{jk} = A_j x_{jk}, \qquad j = 1, 2 \tag{15}$$

$$d_{jk} = c_j x_{jk}, \qquad j = 1, 2 \tag{16}$$

we have the equivalent program, called the extremal program, in the following equations.

Minimize

$$\sum_k d_{1k} \lambda_k + \sum_k d_{2k} \mu_k + \Gamma F \tag{17}$$

subject to

$$\sum_k P_{1k} \lambda_k + \sum_k P_{2k} \mu_k + A_3F = b \tag{18}$$

$$\begin{bmatrix} \sum_k \lambda_k = 1 \\ \sum_k \mu_k = 1 \end{bmatrix} \tag{19}$$

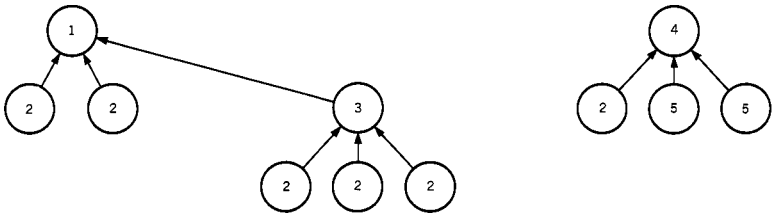
$$\lambda_k, \mu_k, F \geq 0.$$

This is now a program in λ_k, μ_k, F, and if its solution is $\{\lambda_k^0\}$, $\{\mu_k^0\}$, $\mathbf{F}^0$, then the vectors

$$\mathbf{s}_1 = \sum_k \lambda_k^0 x_{1k}, \qquad \mathbf{s}_2 = \sum_k \mu_k^0 x_{2k}, \qquad \mathbf{F}^0 \tag{20}$$

solve the program in the expressions 9 through 12. The solution,

Figure 2 Assembly of products



using Dantzig-Wolfe decomposition, is discussed in Gorenstein,[6] where a "dominance" theorem is proved for this program. The generalized upper bounding method of Dantzig and Van Slyke[7] may also be used. Also see Lasdon and Mackey.[8]

Level-by-level optimization

We proceed to discuss methods where the precedence relations need not be explicitly included in the constraints. Before proceeding to the next suggested method, we will have to define the concept of "level" in an assembly manufacturing process. For a more complete discussion, see Loewner.[9]

If we are considering products such as those shown in Figure 2, we define the concept of distance 1 for item i in relation to item j where item i is needed *directly* to make item j. Thus, item 2 is at distance 1 from item 1, 3, and 4. Similarly, we define distance 2, 3, $\cdots$, and item 2 is *also* at distance 2 from item 1. We then define the level of a part as $d + 1$ where d is the *maximum* distance of the part from a final product; each final assembly is defined as distance zero from itself. This uniquely defines the level of an item. Thus, level 1 consists of items 1, 4; level 2 of items 3 and 5; and level 3 has only item 2. In this manner, all items can be classified into levels—those in the lowest level are called details. An item in a particular level *may* be required for all levels above it but will never be required for levels below its level. It is this property of levels that is used in the next suggested method.

A method of dealing with the precedences and avoiding the problem of integer solutions is to solve the problem level by level. This procedure is suboptimal in that the program is divided into independent subprograms by means of the levels. It automatically preserves the precedence requirements, and therefore, Equation 3 no longer appears explicitly. Thus, there will then be, at most, $2KT$ noninteger solutions for each level; in most applications there are usually no more than five or six levels in the manufacturing process. The procedure is described below.

The demand of the final products is exploded into its item requirements. Resources, work-force bounds, are then allocated to the various levels on a pro-rata basis according to the total demands for the level. Then the program for the first level only is solved using the allocated pro-rata work force as bounds in Equation 6. From this, a solution is obtained for level 1. Then an explosion is made again, this time with the level 1 solution serving as the end products' demands. The remaining available work force (after deducting the level 1 allocation from the first program) is allocated to levels 2 and below, and then the program for level 2 is solved using its allocated work force as the bounds in Equation 6. An explosion follows; this time from level 2 on down using the solution just obtained for level 2 as the level 2 demand. This action continues until the last level of the process is reached.

Since the explosion is done after the optimization at each level, and since the linear program only calls for production earlier than required, we are assured of a precedence-feasible solution in this manner. But, we are not assured that the solution will be optimal unless all parts of the entire problem (Equations 1 through 6) are considered simultaneously.

Explosion within the linear program

We now discuss a method which is similar in concept to the level-by-level optimization, but will lead to an optimal solution and at the same time overcome the problem of integer solutions. This will be accomplished by not explicitly including Equation 3, but by selecting a production schedule for an item on a level-by-level basis in the decomposition subprogram.

We are concerned with the program 1, 2, 4, 5, 6, excluding 3. While Equation 3 is excluded from the specific statement of the program, the restrictions represented will be considered during the course of the solution. We will concern ourselves with the phase II part, since starting the algorithm with phase I is straightforward.

We use the revised simplex method for the program 1, 2, 4, 5, 6 rewritten in matrix form as 9, 10, 11, 12. However, since we exclude the constraints of 3, the elements of A_1 consist only of the $\ell_{ijk\tau}$ and do not include the y_{ijm}, the elements related to the precedences. Thus, A_1 has $2KT$ rows of which the last KT are all zero. Therefore, in the transformation to the extremal program, 17, 18, 19, the vectors $\mathbf{P}_{jk} = \mathbf{A}_j \mathbf{x}_{jk}$ will have $2KT$ components instead of $2KT + P$. The extremal problem thus has a total of $2KT + 2$ rows, compared to the $2KT + 2 + P$ rows in the formulation that included Equation 3.

To perform an iteration of phase II, we have at hand $2KT + 2$ basic variables and their associated columns, which together with their cost components form the basis matrix B^*. We also have the inverse of the basis, B^{*-1}, the first row of which contains the prices (dual variable values), $(1, \pi, \bar{\pi})$. To determine the vector to enter the basis we are led to a subproblem. Solving this program leads either to a vector to enter the basis or to an optimality indication, at which point the program is solved. However, in solving the program, or, more particularly that part of it related to the production schedule variable x_1, we have to introduce a procedure that will preserve the precedences. We wish to make certain that each production plan vector, $\mathbf{P}_{1k}$, to be considered for introduction into the basis, will be precedence feasible, that is, will satisfy the precedence requirements. In order to ensure this, we introduce a level-by-level explosion as we proceed with the item-by-item determination of the solution.

We have the subproblem program

$$\min\ (\pi A_1 - c_1)\, x_1$$

subject to

$$\sum_j x_{1ij} = 1, \qquad i = 1, 2, \cdots, I$$

where

$$x_1 = (x_{111}, \cdots, x_{11J_1}, x_{121}, \cdots, x_{12J_2}, \cdots, x_{1I1}, \cdots, x_{1IJ_I})$$

This involves selecting for each i, the minimum component of

$$(\pi A_1 - c_1)$$

If the minimum is the coefficient of x_{1ij_0}, then x_{1ij_0} is set equal to 1 and the other x_{1ij} equal to 0. Or, schedule j_0 for item i is selected. However, to preserve the precedences, the I items are first placed in level order so that the I items are placed in L classes, with level 1 in class 1, level 2 in class 2, etc. The minimization is first performed for level 1 items. Then using this level 1 solution, an explosion takes place to produce a demand for level 2 items. The dominant level 2 schedules are then computed from this demand, and this set of schedules for a level 2 item will be a subset of the dominant schedules related to an explosion of the original level 1 demand. (Proof of this appears in Gorenstein.[6]) Then, using only this set of schedules for level 2 items, which are precedence-feasible since they result from an explosion of a level 1 solution, perform the minimization for level 2 items. This means, in effect, that the matrix A_1 has had some level 2 columns removed for this iteration, and the minimization is over the set of precedence-feasible columns that have remained. If the minimum for level 2 item i_1 is achieved for schedule j_1, then $x_{1i_1j_1}$ is set equal to 1 and x_{1i_1j} is set equal to 0 for $j \neq j_1$. This is done

for all level 2 items, and the level 2 schedules are procured. Again, an explosion takes place to level 3 items, and this is used as the level 3 demand. We proceed in this manner through all levels. Thus, the solution for the kth iteration, x_1^k, is precedence-feasible, and this is transformed to $\mathbf{P}_{1k}$, the candidate to enter the basis, with cost d_{1k}, by use of Equations 15 and 16.

The production plan solution given by Equation 20,

$$\mathbf{s}_1 = \sum_k \lambda_k^o \, x_{1k}$$

is a convex combination of precedence-feasible solutions and is, therefore, itself precedence-feasible. (See proof in Gorenstein.[6])

In this manner, by performing the minimization over precedence-feasible production plans, we produce at each iteration a candidate production vector to enter the basis which is precedence-feasible. This is done without specifically introducing the constraints of 3, thus enabling us to work with a smaller extremal problem at the expense of having to perform the explosions. But, we have the most important additional advantage—we will have, at most, $2KT$ noninteger solutions for production plans, since the P constraints of Equation 3 are no longer part of the program. Therefore, the basic solution has, at most, $2KT$ noninteger solutions.

Summary

The model of this paper results in a near-optimal solution to a production, inventory, and work-force planning problem where the number of items is large compared to the number of facilities and time periods in the planning horizon. It extends the work of Dzielinski and Gomory[3] to assembly production. Based on the product structures, constraints are generated to maintain the precedence relations among items and solution methods are discussed. Dantzig-Wolfe decomposition is applied to the program and it is noted that generalized upper bounding may also be used. A suboptimal procedure that does a level-by-level optimization is presented, and it may be satisfactory in many applications. The optimal procedure involves a consideration of all levels simultaneously.

REFERENCES

1. A. S. Manne, "Programming of economic lot-sizes," *Management Science* **4,** No. 2, 115–135 (1958).
2. B. P. Dzielinski, C. T. Baker, and A. S. Manne, "Simulation tests of lot-size programming," *Management Science* **9,** No. 2, 229–258 (1963).
3. B. P. Dzielinski and R. E. Gomory, "Optimal programming of lot-sizes, inventory, and labor allocations," *Management Science* **11,** No. 9, 874–890 (1965).

4. G. B. Dantzig and P. Wolfe, "The decomposition algorithm for linear programs," *Econometrica* **29,** No. 4, 767–778 (1961).
5. M. D. Grigoriadis, "Optimal programming of lot-sizes, inventories, and labor allocations—a comment," Communications to the Editor, *Management Science* **12,** No. 7, 622–625 (1966).
6. S. Gorenstein, *Programming for Economic Lot-Sizes with Precedences between Items—An Assembly Model,* Technical Report 320–2995, International Business Machines Corporation, Scientific Center, Philadelphia, Pennsylvania (1970).
7. G. B. Dantzig and R. M. Van Slyke, "Generalized upper bounding techniques," *Journal of Computer and Systems Science* **1,** No. 3, 213–226 (1967).
8. L. S. Lasdon and J. E. Mackey, *An Efficient Algorithm for Multi-Item Scheduling*, SRC 68–9, Systems Research Center, Case Western Reserve University, Cleveland, Ohio (May 1968).
9. P. G. Loewner, "Fabrication and Assembly operations, Part III—Matrix methods for processing configuration data," *IBM Systems Journal* **4,** No. 2, 105–121 (1965).