ON THE JOB-SHOP SCHEDULING PROBLEM*

Alan S. Manne†

Cowles Foundation, Yale University, New Haven, Connecticut

(Received October 19, 1959)

This is a proposal for the application of discrete linear programming to the typical job-shop scheduling problem—one that involves both sequencing restrictions and also noninterference constraints for individual pieces of equipment. Thus far, no attempt has been made to establish the computational feasibility of the approach in the case of large-scale realistic problems. This formulation seems, however, to involve considerably fewer variables than two other recent proposals,[1, 7] and on these grounds may be worth some computer experimentation.

IT WILL BE assumed that this sequencing problem involves the performance of n 'tasks'—each task being defined in such a way as to require the services of a single machine for an integral number of time units. (For concreteness, we may refer to the unit of time as a 'day.') Any one end product will, in general, necessitate the performance of several tasks in sequence. The scheduling problem consists of drawing up a plan for time-phasing the individual jobs so as to satisfy: (a) sequencing requirements—e.g., the children must be washed before they are dried, and (b) equipment interference problems—e.g., the one-year-old and the three-year-old cannot occupy the bathtub at the same time. (All parents will devoutly hope that each of *these* tasks can be performed in less than a day.)

The integer-valued unknowns x_j are to indicate the day on which task j is to be begun ($x_j = 0, 1, \cdots, T$).‡ Just as in Selmer Johnson's formulation,[6] the schedule is to be drawn up so as to minimize the 'make-span,' i.e., the elapsed calendar time for the performance of all jobs—subject, of course, to the constraints upon sequencing and machine interference, and also subject to any delivery date requirements on individual items.

Noninterference Restrictions

Suppose that jobs j and k require a_j and a_k consecutive days respec-

* Research undertaken by the Cowles Commission for Research in Economics under Contract Nonr-358(01), NR 047-066 with the Office of Naval Research.

† This paper was stimulated by a paper of E. H. BOWMAN.[1] Bowman's ideas have led directly to the model formulated here.

‡ For purposes that will shortly become evident, it will be convenient to suppose that we have sufficient a priori knowledge to be able to select a (large) integer T that will constitute a *redundant* upper bound upon the unknowns x_j.

tively.§ Then if they are to be prevented from occupying the same machine at the same time, we must require that one of the two must precede the other by sufficient time so that the first one can be completed before the second is begun: either

$$x_j - x_k \geqq a_k, \quad \text{or else} \quad x_k - x_j \geqq a_j. \tag{1}$$

In order to convert this condition into a linear inequality in integer unknowns, it will be convenient to define a new integer-valued variable y_{jk}, and to write down the following restrictions:

$$0 \leqq y_{jk} \leqq 1, \tag{2}$$

$$(T + a_k)y_{jk} + (x_j - x_k) \geqq a_k, \tag{3}$$

$$(T + a_j)(1 - y_{jk}) + (x_k - x_j) \geqq a_j. \tag{4}$$

Condition (2) ensures that y_{jk} equals either zero or else unity. We already know that $|x_j - x_k| \leqq T$. The effect of conditions (3) and (4) may therefore be summarized as follows: If

$$(x_j - x_k)\begin{Bmatrix} >0 \\ =0 \\ <0 \end{Bmatrix}, \quad \text{then} \quad y_{jk} = \begin{Bmatrix} 0, 1 \\ 1 \\ 1 \end{Bmatrix} \quad \text{and} \quad y_{jk} = \begin{Bmatrix} 0 \\ 0 \\ 0, 1 \end{Bmatrix},$$

where the first set of values for y_{jk} is implied by condition (3) and the second set by condition (4).

Hence if $(x_j - x_k) = 0$, there is no value that can be assigned to y_{jk} so as to satisfy both (3) and (4). If, on the other hand, $(x_j - x_k) \neq 0$, y_{jk} will be set at a value of either zero or unity depending upon which job is to precede the other. Equations (3) and (4) then ensure that the first job will be initiated in sufficient time to be completed before the beginning of the second one. Note that with the classical form of linear programming, it would have been impossible to specify such an either-or condition as (1). This noninterference restriction leads directly to a nonconvex set of restraints upon the unknowns. It is little wonder that Gomory's discovery of integer programming[4] has led to a revival of interest in the machine interference problem.

Sequencing Restrictions

Once the noninterference stipulations have been written down, the remainder of the formulation becomes virtually automatic. If job j is to precede k, this means that job k is to be performed at least a_j days later than j. The integer programming condition becomes:

$$x_j + a_j \leqq x_k. \tag{5a}$$

§ In a somewhat more complex model, one could allow for the possibility of 'hereditary' effects—i.e., dependence of the processing time of one item upon the machine setting employed for the previous one.

'Weak' precedence relations may be written in an analogous fashion. For example, in order to specify that both jobs i and j precede k, but that there is no precedence restriction affecting the performance of i and j, we would have:

$$x_i+a_i\leqq x_k, \qquad x_j+a_j\leqq x_k. \tag{5b}$$

Still another possibility might be that there be a delay of exactly Θ_{jk} days between the performance of jobs j and k. Such a restriction would be indicated by:

$$x_j+a_j+\Theta_{jk}=x_k. \tag{5c}$$

Specific Delivery Requirements

It may happen that the shop is committed to the delivery of an individual job no later than a specified date. If task j is the last task which the shop is to perform upon the item, and if the item is to be available on day d_j, this form of requirement may be written:

$$x_j+a_j\leqq d_j. \tag{6}$$

Over-all Delivery Requirements

Following JOHNSON,[6] we shall employ as our minimand the 'make-span' or total calendar time needed for the performance of all prospective jobs. If this calendar time is denoted by t, the problem now consists of the minimization of t with respect to the nonnegative integers x_j and y_{jk}, subject to constraints (2) to (6), and also subject to:

$$x_j+a_j\leqq t. \qquad (j=1, \cdots, n) \tag{7}$$

The economist, conditioned as he is to take a dim view of any minimand other than dollar costs, will find it difficult to be altogether happy with Johnson's criterion, the minimization of t, the make-span. In defending this choice of minimand, however, it should be pointed out that t is likely to be correlated with dollar costs. In minimizing t we may conceivably also obtain the following cost and profit benefits: (a) a lowered lowered amount of inventory tied up in work-in-process, (b) a shorter *average* customer delay time, and (c) a lower amount of idle time incurred prior to the performance of all currently booked jobs—i.e., a greater capacity to take on additional work as new orders materialize. To the extent that all of these factors work in the same direction, calendar time might constitute a reasonable proxy variable for economic cost. The job sequence that serves to minimize the make-span might also be one that scores quite well on the criterion of dollar costs.

Computational Aspects

Excluding all of the slack variables and also the minimand t, the number of unknowns here is equal to the total number of the x_j plus the y_{jk}.

If, then, there are n tasks and if also there are m possible conflicting pairs of machine assignments, the total number of unknowns would come to $n+m$. For example, with 5 machines and with 10 tasks to be performed on each machine, we would have $n=50$, and $m=\frac{1}{2}(5)(10)(10-1)=225$. The total number of integer-valued unknowns x_j and y_{jk} would come, therefore, to 275—an impressive computational load but by no means an impossible one.*

It is worth pointing out that if an algorithm were available to handle 'mixed' integer programming problems—problems in which some of the unknowns are constrained to take on integer values and others are permitted to be continuous—this scheduling model would fit very naturally into the category of such a 'mixed' problem. The y_{jk} unknowns here are necessarily of a discrete nature. [Otherwise, it would be impossible to impose condition (1)]. However, it might be more efficient and possibly more realistic to regard the start-dates x_j as continuous variables, and not to constrain these to be integers. The success of this kind of modification would hinge entirely upon the computational costs of any algorithms designed to handle 'mixed' problems.

In further work along these lines, one of the most important avenues to be explored would be the possibility of reducing the number of unknowns y_{jk}. Aside from the upper bound constraints (2), these unknowns are involved only in connection with the machine-interference conditions (3) and (4). Since many of these restrictions will inevitably turn out to be redundant in any particular numerical problem, it might be quite feasible to apply here a computer code designed around DANTZIG's principle of 'secondary constraints.'†[3] In the traveling-salesman problem, for example, one does not write down explicitly all conceivable 'loop constraints,'

* By contrast, using the formulation discussed in Part II of WAGNER's paper,[7] the total number of unknowns would come to 600—again neglecting slack variables and also the make-span minimand. In general, Wagner's formulation will require slightly more than twice the number of unknowns in the current proposal. (Because of the large number of inequalities implied by his condition (8), the current proposal is bound to be advantageous in terms of the number of inequality restraints.)

Why concentrate upon the number of nonslack variables, and why ignore the number of constraints in estimating computation requirements? Because with Gomory's method for integer programming, one is repeatedly applying the dual simplex algorithm—an algorithm in which the original number of nonbasis variables is apparently the critical factor. Needless to say, with the limited amount of computing experience currently available in the area of integer programming, one cannot afford to be too dogmatic with such predictions.

One further comment: I feel that the real advantage of Wagner's formulation over this one lies in the case where it is known that the order position of each end product is the same from one machine in the processing sequence to the next (Part IV of his paper). This is a really important case, and I suspect that his approach will turn out to be quite practical there.

† RICHARD LEVITAN is in the process of writing an integer-programming computer code in which this principle is being applied.

but only those that have been violated during the course of previous iterations.[2] By applying the identical principle to the machine-scheduling problem, this suggestion may conceivably make it economical to obtain specific solutions to realistic examples.

In weighing the benefits from exact optimization of large systems via integer programming, the investigator ought not to permit himself to ignore the likelihood that Monte Carlo methods for *approximate* optimization will entail lower computing costs. A very promising start along these lines has recently been reported by HELLER.[5]

REFERENCES

1. E. H. BOWMAN, "The Schedule-Sequencing Problem," *Opns. Res.* **7,** 621–624 (1959).
2. G. B. DANTZIG, ET AL., "Solution of a Large-Scale Traveling-Salesman Problem," *Opns. Res.* **2,** 393–410 (1954).
3. ———, "Upper Bounds, Secondary Constraints, and Block Triangularity in Linear Programming," *Econometrica* (April 1955).
4. R. GOMORY, "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bulletin of the American Mathematical Society* (September 1958).
5. J. HELLER, "Some Numerical Experiments for an $M \times J$ Flow Shop and Its Decision-Theoretical Aspects," *Opns, Res.* **8,** 178–184 (1960).
   New York University, February 1, 1959, unpublished mimeographed report.
6. S. M. JOHNSON, "Optimal Two and Three Stage Production Schedules and Setup Times Included," *Naval Res. Log. Quart.* (March 1954).
7. H. WAGNER, "An Integer Linear-Programming Model for Machine Scheduling," *Naval Res. Log. Quart.* (June 1959).