A Functional Equation and Its Application to Resource Allocation and Sequencing Problems
Author(s): E. L. Lawler and J. M. Moore
Source: *Management Science*, Vol. 16, No. 1, Theory Series (Sep., 1969), pp. 77-84
Published by: INFORMS
Stable URL: http://www.jstor.org/stable/2628367
Accessed: 23-06-2017 22:58 UTC

MANAGEMENT SCIENCE
Vol. 16, No. 1, September, 1969
Printed in U.S.A.

A FUNCTIONAL EQUATION AND ITS APPLICATION TO RESOURCE ALLOCATION AND SEQUENCING PROBLEMS*,**

E. L. LAWLER[1] AND J. M. MOORE[2]

A functional equation, similar to that formulated for the knapsack problem, is applied to the solution of a problem of resource allocation in critical path scheduling, and to a variety of single-machine sequencing problems with deadlines and loss functions. Among these are: minimization of the weighted number of tardy jobs, maximization of weighted earliness, minimization of tardiness with respect to common relative and absolute deadlines, and minimization of weighted tardiness with respect to a common deadline. Extensions to multiple-machine sequencing problems are discussed, and the sequencing of two machines in series is treated in detail.

1. Introduction

Suppose n jobs are to be performed one at a time in the fixed order $1, 2, \cdots, n$. Each job can be performed in either one of two different modes. The processing of job j requires a_j units of time in one mode and b_j units in the other. In the first mode, a loss of $\alpha_j(t)$ units is incurred upon the completion of the job at time t, and $\beta_j(t)$ units in the other. What assignment of modes to jobs and what timing of the jobs will minimize the total loss?

Let $f(j, t)$ = the minimum total loss for the first j jobs, subject to the constraint that job j is completed no later than time t.

By the usual dynamic programming argumentation:

$$f(0, t) = 0, \qquad (t \geqq 0),$$

$$f(j, t) = +\infty, \qquad (j = 0, 1, \cdots, n; t < 0),$$

$$(1) \qquad f(j, t) = \min\begin{Bmatrix} f(j, t-1), \\ \alpha_j(t) + f(j-1, t-a_j), \\ \beta_j(t) + f(j-1, t-b_j) \end{Bmatrix}, \qquad (j = 1, 2, \cdots, n;\ t \geqq 0).$$

The problem is solved by the calculation of $f(n, T)$, where T is a sufficiently large number. For example, if all of the α_j's and β_j's are monotone nondecreasing, we may choose

$$T = \textstyle\sum_{j=1}^{n} \max\{a_j, b_j\}.$$

The overall computation requires on the order of nT computational steps.

We shall show that the functional equation (1) can be used to solve a variety of sequencing and scheduling problems, including several which do not at all involve a fixed order for the processing of jobs.

The computations for these sequencing problems are vastly more efficient than, for example, the computational methods of Held and Karp [1], Schild and Fredman [9],

* Received January 1968 and revised January 1969.

** Some of the material in this paper was presented at the 34th National ORSA Meeting, Philadelphia, November 1968, under the title, "Optimal Sequencing of Jobs Subject to Deadlines."

[1] The University of Michigan, currently visiting the University of California at Berkeley.

[2] Esso Mathematics and Systems, Incorporated.

or Lawler [4]. The reason for this is that those methods carry out an implicit search over n ! possibilities, whereas ours is a search over only 2^n.

2. Sequencing with Deadlines and Precedence Constraints

In order to illustrate the initial application of Equation (1), we must first solve the following problem. Suppose n jobs are to be processed by a single machine. Job j requires a_j units of processing time and it has a deadline d_j for its completion. There are certain precedence constraints which must be satisfied, and these are expressed in the form of a partial ordering relation "ρ" on the jobs; if $i\rho j$, then job i must precede j. Does there exist a sequence for the jobs, consistent with the given precedence constraints, such that each job is completed by its dealine?

Assume, without loss of generality, that the jobs are numbered in such a way that $i\rho j$ implies $i \leqq j$. (There are a number of possible schemes for obtaining such a numbering.) Now, for $j = 1, 2, \cdots, n$, let

$$\bar{d}_j = \min \{d_k \mid j\rho k\} + j\epsilon,$$

where ϵ is a small number and $j\epsilon$ is used to break ties. It is important to note that $i\rho j$ implies $\bar{d}_i < \bar{d}_j$.

Theorem: Each job can be completed on time, consistent with the given precedence constraints, if and only if each job is completed on time in the sequence obtained by ordering jobs according to increasing values of $\bar{d}_j$.

Proof: The "if" part is obvious, so we consider the situation in which there exists a hypothetical sequence in which all jobs are completed on time, but in which the jobs are not ordered according to increasing values of $\bar{d}_j$. Let i, j be a consecutive pair of jobs in this hypothetical sequence, with $\bar{d}_i > \bar{d}_j$. It cannot be the case that $i\rho j$, because that would imply $\bar{d}_i < \bar{d}_j$, contrary to assumption. Hence, if we interchange the positions of i and j, we do not violate the precedence constraints.

A little further analysis reveals that either j, or some successor of j, has a dealine at least as early as that of i. Since j, or its successor, is on time in the given sequence, placing i in the position of j will maintain the on-time status of i. And, of course, if i and j are interchanged, j will remain on time since it will be earlier in the sequence.

Thus, i and j can be interchanged in the sequence. And, by a finite number of such interchanges, the hypothetical sequence can be transformed into one in which the jobs are placed in order of increasing values of $\bar{d}_j$, with all jobs on time, and with the precedence constraints satisfied. *Q.E.D.*

This theorem generalizes the result of Jackson [2], reported by Smith [10], to the effect that an unrestricted set of jobs can all be completed on time if and only if they are completed on time when sequenced according to deadlines, earliest first. Operationally, the generalized rule can be stated as follows: When operating a machine, always take next from among the jobs which are currently available (*i.e.*, jobs none of whose predecessors remain unprocessed), a job which has a successor with the earliest possible deadline (considering a job to be one of its own successors).

It is particularly important to note that the sequence specified by the theorem is completely independent of processing times. This fact makes possible the solution method described in the next section.

3. Resource Allocation in Critical Path Scheduling

Consider a project which is composed of a large number of tasks related by a complex set of precedence constraints. A critical path analysis is employed to determine a deadline for each individual task. Suppose a certain subset of tasks can be performed only one at a time (perhaps because of the limited availability of specialized personnel or equipment). For each task j in the subset, there is a choice between a long processing time a_j at low cost α_j or a short processing time b_j at high cost β_j. How should the tasks be processed with least cost so that all deadlines are met?

The answer is to order the jobs according to the generalized dealine rule given in the theorem, and then to employ Equation (1) in a straightforward way. This requires on the order of nd_n computational steps.

4. Application to Single-Machine Sequencing Problems

In the next several sections, we describe the application of Equation (1) to a number of single-machine sequencing problems involving deadlines and loss functions. The essential problem formulation is as follows. A set of n jobs are to be performed in sequence—one immediately following the other—by a single machine. Job j requires a'_j units of processing time, and a loss of $c_j(t)$ is incurred for its completion at time t. The problem is to find a permutation π (where $\pi(j) = k$ if job j is the k^{th} to be performed) such that

$$\sum_{j=1}^{n} c_j(t_j)$$

is minimized, where

$$t_j = \sum_{k=1}^{\pi(j)} a'_{\pi^{-1}(k)}$$

is the time of completion of job j.

This is essentially the general problem which Held and Karp [1] solved with on the order of $n2^n$ computational steps. By contrast, we shall deal with a number of special instances that can be solved with algebraic, rather than exponential, growth in the number of computational steps.

The trick in all of these applications is to notice that, for certain types of loss functions $c_j(t)$, the jobs can be partitioned into two classes. One class of jobs will be performed in a predetermined order, and the other in an arbitrary order, either preceding or following the first class. The actual assignment of jobs to classes is determined by the solution of Equation (1) with "modes" corresponding to classes.

5. Minimization of Weighted Number of Tardy Jobs

Suppose the loss functions $c_j(t)$ are of the form

$$\begin{aligned} c_j(t) &= 0, && (t \leqq d_j), \\ &= pj, && (t > d_j). \end{aligned}$$

That is, a penalty p_j is exacted if job j is completed later than its assigned deadline d_j. How should the jobs be sequenced so as to minimize the total loss, *i.e.*, the weighted number of tardy jobs?

Following the reasoning of Moore [6] (who dealt with the special case $p_j = 1$) and Rothkopf [8], we note that the problem effectively requires us to partition the jobs into two classes: those which are to be on time, and those which are to be tardy. We can assume that the on-time jobs will be sequenced in order of their deadlines with the tardy jobs following them in arbitrary order.

The problem is solved by ordering the jobs by deadlines, earliest deadline first, and then applying Equation (1) with

$$a_j = a_j', \qquad \alpha_j(t) = 0$$

$$b_j = 0, \qquad \beta_j(t) = p_j.$$

Whichever jobs are given zero processing times in the solution of (1) will be deemed to be tardy jobs.

6. Relation to Knapsack Problem

The reader may already have been struck by the resemblance between Equation (1) and functional equations which are used to solve the knapsack problem.

Using consistent notation, the knapsack problem is:

$$\begin{aligned}
&\text{Maximize } \textstyle\sum_{j=1}^{n} p_j x_j \\
&\text{Subject to } a'_1 x_1 + a'_2 x_2 + \cdots + a'_n x_n \leqq d, \\
&\qquad\qquad x_j = 0 \quad \text{or} \quad 1 \; (j = 1, 2, \cdots, n).
\end{aligned}$$

It can be solved by the equation

$$f(j,t) = \max \begin{Bmatrix} f(j, t-1), \\ f(j-1, t), \\ p_j + f(j-1, t-a'_j) \end{Bmatrix}, \qquad (t \leqq d), \tag{2}$$

$$= f(j, d), \qquad (t > d).$$

A little manipulation shows that the problem in the previous section is equivalent to:

$$\begin{aligned}
&\text{Maximize } \textstyle\sum_{j=1}^{n} p_j x_j \\
&\text{Subject to } a'_1 x_1 && \leqq d_1 \\
&\qquad a'_1 x_1 + a'_2 x_2 && \leqq d_2 \\
&\qquad a'_1 x_1 + a'_2 x_2 + a'_3 x_3 && \leqq d_3 \\
&\qquad\qquad \vdots \\
&\qquad a'_1 x_1 + a'_2 x_2 + \cdots + a'_n x_n && \leqq d_n
\end{aligned}$$

where

$$\begin{aligned}
x_j &= 0 \quad \text{if job } j \text{ is to be tardy,} \\
&= 1 \quad \text{if job } j \text{ is to be on time.}
\end{aligned}$$

For this problem, Equation (1) becomes equivalent to

$$f(j,t) = \max \begin{Bmatrix} f(j, t-1), \\ f(j-1, t), \\ p_j + f(j-1, t-a'_j) \end{Bmatrix}, \qquad (t \leqq d_j) \tag{3}$$

$$= f(j, d_j). \qquad (t > d_j)$$

This problem formulation and method of solution were previously noted by Rothkopf [8].

Note: In both (2) and (3) above, the term $f(j, t-1)$ is dominated by the other two alternatives and can be eliminated; this term has been included so as to point up the correspondence to (1).

7. Linear Loss Functions, Common Deadline

Again suppose n jobs are to be processed on a single machine, with loss functions of the form

$$\begin{aligned} c_j(t) &= p_j t, \qquad (t \leqq d), \\ &= q_j, \qquad (t > d), \end{aligned}$$

where d is a deadline common to all jobs. How should the jobs be sequenced so as to minimize the total loss?

The problem effectively requires us to partition the jobs into two classes: those which are completed on or before the deadline, and those which are tardy. The on-time jobs will be sequenced according to the ratios p_j/a'_j (see Smith [10] and McNaughton [5]), followed by the tardy jobs in arbitrary order.

The problem is solved by ordering the jobs by the ratios p_j/a_j', the job with the largest ratio first, and applying Equation (1) with

$$\begin{aligned} a_j &= a'_j, \qquad \alpha_j(t) = p_j t \\ b_j &= 0, \qquad \beta_j(t) = q_j. \end{aligned}$$

The calculation of $f(n, d)$ yields the desired solution with on the order of nd computational steps.

8. Maximization of Weighted Earliness

Let

$$\begin{aligned} c_j(t) &= p_j t, \qquad (t \leqq d), \\ &= q_j, \qquad (t > d), \end{aligned}$$

as in the previous section, and suppose that $p_j d \geqq q_j$ for all j. Then a sequence which is minimal with respect to the functions $c_j(t)$ is also minimal with respect to the functions

$$c'_j(t) = \min \{p_j t, q_j\}.$$

But a sequence which is minimal with respect to the functions $c'_j(t)$ is also maximal with respect to the functions

$$c''_j(t) = \max \{p_j(d_j - t), 0\},$$

where $d_j = q_j/p_j$.

We note that $c''_j(t)$ represents weighted earliness of the job j with respect to an individual dealine d_j. Thus, the methods of Section 7 can be used to maximize weighted earliness with respect to individual deadlines, or, for that matter, to maximize weighted tardiness. However, the problem of *minimizing* weighted tardiness is a much more difficult matter, as we shall see below.

9. Minimization of Tardiness, Common Relative and Absolute Deadlines

Consider the same problem, but this time let

$$\begin{aligned} c_j(t) &= 0, \qquad (t \leqq d), \\ &= t - d, \qquad (d \leqq t \leqq d'), \\ &= q_j, \qquad (t > d'), \end{aligned}$$

where d and d' are two deadlines common to all jobs. We may view d as a "relative" and d' as an "absolute" deadline.

The problem requires us to partition the jobs into two classes: those which are on time with respect to d', and those which are tardy. The on-time jobs can be sequenced according to the order of their processing times, shortest processing time first. (See Smith [10].)

The problem is solved by ordering the jobs by the shortest processing time rule, and then applying Equation (1) with

$$a_j = a'_j, \qquad \alpha_j(t) = \max\{0, t - d\},$$
$$b_j = 0, \qquad \beta_j(t) = q_j.$$

The calculation of $f(n, d)$ yields the desired solution.

10. Minimization of Weighted Tardiness, Common Deadline

Consider the same problem, but let

$$c_j(t) = \max\{0, p_j(t - d)\}.$$

In other words, $c_j(t)$ represents weighted tardiness with respect to a common deadline d.

This problem requires a partitioning of the jobs into three classes: (A) those which are completed on time, which can be sequenced in arbitrary order, (B) those which are late, and which are sequenced in order of the ratios p_j/a_j', largest ratio first and (C) a single job which begins before the deadline d and is completed on or after it.

There appears to be no way to know, a priori, the identity of the job in Class (C), except that it must possess a p_j no greater than that of any job in Class (A). Hence, it seems just as well to resolve the problem for each possible job in Class (C) and choose the best result.

The procedure is as follows:

(1) Order the jobs according to increasing ratios p_j/a'_j, *i.e.*, $p_1/a'_1 \leqq p_2/a'_2 \leqq \cdots \leqq p_n/a'_n$.

(2) Solve Equation (1) for each subset of $n - 1$ jobs, with

$$a_j = a'_j, \qquad \alpha_j(t) = 0,$$
$$b_j = 0, \qquad \beta_j(t) = p_j t.$$

Let $f^{(k)}(n, t)$ denote the solution for the subset of jobs $\{1, 2, \cdots, k-1, k+1, \cdots, n\}$.

(3) Find the values of k and t for which

$$f^{(k)}(n, t) + p_k(t + a'_k - d), \qquad (d - a'_k \leqq t < d),$$

is minimum.

These values indicate the identity of the job in Class (C) and its time of completion, while $f^{(k)}(n, t)$ can be used to determine the jobs in Classes (A) and (B). The overall computation requires on the order of n^2d steps.

In order to justify the expression for $\beta_j(t)$, the reader should consider the partition of the first j jobs into the two classes (A) and (B). Let

$$A_j = \sum_{i=1}^{j} a'_i.$$

Then if the total processing time of the jobs in Class (A) is t, the total processing time of the jobs in Class (B) is $A_j - t$. It follows that if job j is assigned to Class (B) its completion time will be $A_n - (A_j - t)$, and the loss incurred will be

$$p_j(A_n - (A_j - t)) = p_j t + \text{constant}.$$

11. Machines in Parallel

Each of the problem formulations and solution methods we have discussed can be extended in a very natural way to the situation in which there are m machines which can process jobs simultaneously. For machine i, job j can be processed either in time $a_j^{(i)}$ or $b_j^{(i)}$, with losses $\alpha_j^{(i)}(t)$ and $\beta_j^{(i)}(t)$.

It is important to note that the jobs which are assigned to any given machine are processed in an order which is consistent with the ordering that would be used in the related single machine problem. This enables us to define a function

$f(j, t_1, t_2, \cdots, t_m) =$ the minimum total loss for the first j jobs, subject to the constraint that no job is completed later than time t_i on machine i,

and to obtain the appropriate functional equation.

In these cases, the number of computational steps grows as mnT^m, where T is an appropriately large number. Some reduction of computational complexity is possible because of symmetry in the case that the machines are identical (processing times and loss functions the same). One can then assume that $t_1 \leqq t_2 \leqq \cdots \leqq t_n$. Nevertheless, the computation still grows essentially exponentially. The reader is referred to Rothkopf [7, 8] for further details.

12. Two Machines in Series

Suppose n jobs are to be performed by each of two machines in sequence. That is, job j is to be performed by the first machine with processing time $a_j^{(1)}$ and then by the second with processing time $a_j^{(2)}$. The jobs are subject to a common deadline d. If job j is completed by the second machine at time t, a loss of

$$\begin{aligned} c_j(t) &= 0, \qquad (t \leqq d), \\ &= p_j, \qquad (t > d), \end{aligned}$$

is incurred. How should the jobs be sequenced so as to minimize the total loss?

Again the problem is to partition the jobs into two classes: those which are to be on time, and those which are to be late. The on-time jobs can be assumed to be sequenced according to the rule of S. Johnson [3], and these jobs can be followed by the tardy jobs, in arbitrary order.

Assume that the jobs are numbered according to Johnson's rule:

$$\min \{a_j^{(1)}, a_{j+1}^{(2)}\} \leqq \min \{a_{j+1}^{(1)}, a_j^{(2)}\}$$

then let

$f(j, t_1, t_2) =$ the minimum total loss for the first j jobs, subject to the constraint that no job is completed later than time t_1 by the first machine and time t_2 by the second.

By arguments similar to those used above,

$$f(0, t_1, t_2) = 0, \qquad (t_1, t_2 \geqq 0),$$

$$f(j, t_1, t_2) = +\infty, \qquad (j = 0, 1, \cdots, n; t_1 < 0 \text{ or } t_2 < 0),$$

$$f(j, t_1, t_2) = \min \begin{Bmatrix} f(j, t_1, t_2 - 1), \\ f(j, t_1 - 1, t_2), \\ p_j + f(j - 1, t_1, t_2), \\ f(j - 1, t_1 - \delta, t_2 - a_j^{(2)}) \end{Bmatrix}, \quad (j = 1, 2, \cdots, n; t_1, t_2 \geqq 0),$$

where $\delta = \min \{t_1 - a_j^{(1)}, t_2 - a_j^{(1)} - a_j^{(2)}\}$.

An optimal solution is obtained by computing $f(n, d, d)$, and this requires on the order of $n\,d^2$ computational steps.

Acknowledgement

Support for the first author was provided under Air Force Contract AF 30(602)-3546 at the Systems Engineering Laboratory, The University of Michigan, Ann Arbor. Research was performed while the second author was affiliated with the Department of Industrial Engineering, The University of Michigan, Ann Arbor.

References

1. HELD, MICHAEL AND RICHARD M. KARP, "A Dynamic Programming Approach to Sequencing Problems," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 10, No. 1, pp. 196–210, (March 1962).
2. JACKSON, J. R., "Scheduling a Production Line to Minimize Maximum Tardiness," Management Science Research Project, Report No. 43, University of California, Los Angeles, (July 1955).
3. JOHNSON, S. M., "Optimal Two- and Three- Stage Production Schedules with Set-Up Times Included," *Naval Research Logistics Quarterly*, Vol. 1, No. 1, pp. 61–68, (1954).
4. LAWLER, EUGENE L., "On Scheduling Problems with Deferral Costs," *Management Science*, Vol. 11, No. 2, pp. 280–288, (November 1964).
5. MCNAUGHTON, ROBERT, "Scheduling with Deadlines and Loss Functions," *Management Science*, Vol. 6, No. 1, pp. 1–12, (October 1959).
6. MOORE, J. M., "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," *Management Science*, Vol. 15, No. 1, pp. 102–109, (September 1968).
7. ROTHKOPF, MICHAEL H., "Scheduling Independent Tasks on Parallel Processors," *Management Science*, Vol. 12, No. 6, pp. 437–447, (January 1966).
8. ROTHKOPF, MICHAEL H., "Scheduling Independent Tasks on One or More Processors," Ph.D. Thesis, M.I.T. School of Industrial Management, (January 1964). Also available as Interim Technical Report No. 2, Operations Research Center, Massachusetts Institute of Technology.
9. SCHILD, A. AND I. J. FREDMAN, "Scheduling Tasks with Deadlines and Nonlinear Loss Functions," *Management Science*, Vol. 9, No. 1, pp. 73–81, (October 1962).
10. SMITH, WAYNE E., "Various Optimizers for Single-Stage Production," *Naval Research Logistics Quarterly*, Vol. 3, Nos. 1–2, pp. 59–66, (March–June 1956).