OPTIMAL SEQUENCING OF A SINGLE MACHINE SUBJECT TO PRECEDENCE CONSTRAINTS

by

E. L. Lawler

Memorandum No. ERL-M327

20 August 1971

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Optimal Sequencing of a Single Machine Subject to Precedence Constraints

E. L. Lawler

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory,
University of California, Berkeley, California 94720

Abstract

Suppose n jobs are each to be processed by a single machine, subject to arbitrary given precedence constraints. Associated with each job j is a known processing time a_j and a monotone nondecreasing cost function $c_j(t)$, giving the cost that is incurred by the completion of that job at time t. The problem is to find a sequence which will minimize the maximum of the incurred costs. An efficient computational procedure is given for this problem, generalizing and simplifying previous results of the present author and J. M. Moore.

Research sponsored by the U.S. Naval Electronic Systems Command, Contract N00039-71-0255.

1. Problem Formulation

Suppose n jobs are each to be processed by a single machine (one at a time, with no interruptions), subject to arbitrary given precedence constraints. Associated with each job j is a known processing time a_j and a monotone nondecreasing cost function $c_j(t)$, giving the cost that is incurred by the completion of that job at time t. The problem is to find a sequence which will minimize the maximum of the incurred costs.

2. Sequencing Theorem

Theorem Let S denote the subset of jobs which may be performed last, i.e. those jobs which are not required to precede any others. Let T denote the sum of the processing times of all the jobs. Let k be a job in S such that

$$c_k(T) = \min_{j \in S}\{c_j(T)\}.$$

Then there exists a minmax optimal sequence, i.e. a sequence which minimizes the maximum of the incurred costs, in which job k is last.

Proof:

Consider any sequence π' with job $k' \neq k$ last. Such a sequence can be represented schematically as follows, where A and B represent the portions of the sequence occupied by the n-2 jobs other than k and k':

π': | A | k | B | k' |

Suppose we modify this sequence to obtain:

π: | A | B | k' | k |

If the given precedence constraints are observed by sequence π', then they are also observed by sequence π, since neither k nor k' is required to precede any other job.

No job is completed later in π than in π', except job k. Because the cost functions are assumed to be monotone, it follows that no incurred cost can be greater in π than in π', except possibly that of k. But job k was chosen to be such that $c_k(T) \leq c_{k'}(T)$. Hence it follows that the maximum of the incurred costs is no greater for sequence π than for π'.

3. Sequencing Algorithm

An efficient algorithm for finding a minmax optimal sequence follows immediately from the theorem above. One simply finds a job k which can be placed last. Then, having removed k from the problem, one finds a job which can be placed last among the remaining n-1 jobs and second-to-last in the complete sequence, and so on.

We illustrate the procedure with a simple example.

Suppose there are four jobs, with precedence constraints as indicated in Figure 1. There jobs have processing times $a_1=1$, $a_2=2$, $a_3=2$, $a_4=1$, and cost functions as indicated in Figure 2.

Jobs 2 and 4 are eligible to be processed last in an optimal sequence, since they have no successors, as seen in Figure 1. Whatever job is last in the sequence will be completed at time t=6. We compare the values $c_2(6)$ and $c_4(6)$, marked by "x's" in Figure 2, and find that $c_4(6) < c_2(6)$. Accordingly, we place job 4 in the last position of the optimal sequence.

If job 4 is last in the sequence, jobs 2 and 3 are both eligible to

be processed in the next-to-last position. Whatever job is processed in this position in the sequence will be completed at time t=5. We compare the values of $c_2(5)$ and $c_3(5)$, and find that $c_3(5) < c_2(5)$. Accordingly, we place job 3 in the next-to-last position of the optimal sequence.

There is only one way to order the remaining jobs 1 and 2. Hence a minmax optimal sequence is 1,2,3,4. The maximum cost is incurred by the completion of job 3 at t=5.

4. Computational Efficiency

In reference [1], J. M. Moore suggests a procedure for finding a minmax optimal sequence in the case that there are no precedence constraints. The present procedure applies to the more general problem with precedence constraints, and is, in addition, more efficient computationally. It can be shown that the present procedure requires a number of computational steps which grows as n^2, where n is the number of jobs.

5. Sequencing with Deadlines and Precedence Constraints

Suppose, instead of a cost function $c_j(t)$, there is simply specified for each job a deadline d_j. The problem is to find a sequence, subject to the given precedence constraints, which will cause each job to be completed by its deadline, if such a sequence exists.

This problem was solved in reference [2] by a method in which each job j was assigned a new deadline $\bar{d}_j \leq d_j$, and the jobs were sequenced according to increasing values of $\bar{d}_j$. It was noted that the solution to this problem was independent of the processing times of the jobs.

The computation also suggested the following "first-to-last" rule:

Sequence the jobs from first to last, always choosing next from among the jobs which are currently available (i.e. jobs none of whose predecessors remain unchosen), a job which has a successor with the earliest possible deadline (considering a job to be one of its own successors).

This problem can also be analyzed as a minmax problem as follows. Assign to each job j a cost function $c_j(t)$ as shown in Figure 3. By following the algorithm of Section B, we discover that a minmax optimal sequence is obtained by the following "last-to-first" rule, which is a generalization of the well-known "deadline rule" for optimal sequencing without precedence constraints.

Sequence the jobs from last to first, always choosing next, from among the jobs which are currently available (i.e. jobs none of whose successors remain unchosen) a job with the latest possible deadline.

As before, we note that this sequence is independent of the actual processing times. The processing times are used to check to see if the sequence does in fact avoid tardiness of all of the jobs. If the sequence is not successful in this regard, no other sequence is.

Finally, we note that the sequence obtained by either of these two rules serves to minimize the maximum tardiness (or lateness) of the jobs, if it is not possible to complete all the jobs on time.

References

[1] J. M. Moore, "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," Mgt. Sci., 15, 102-109 (1968).

[2] E. L. Lawler and J. M. Moore, "A Functional Equation and Its Application to Resource Allocation and Sequencing Problems," Mgt. Sci., 16, 77-84 (1969).

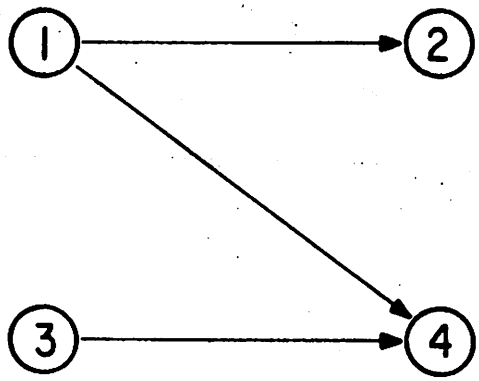

Fig. 1. Precedence Constraints for Example.

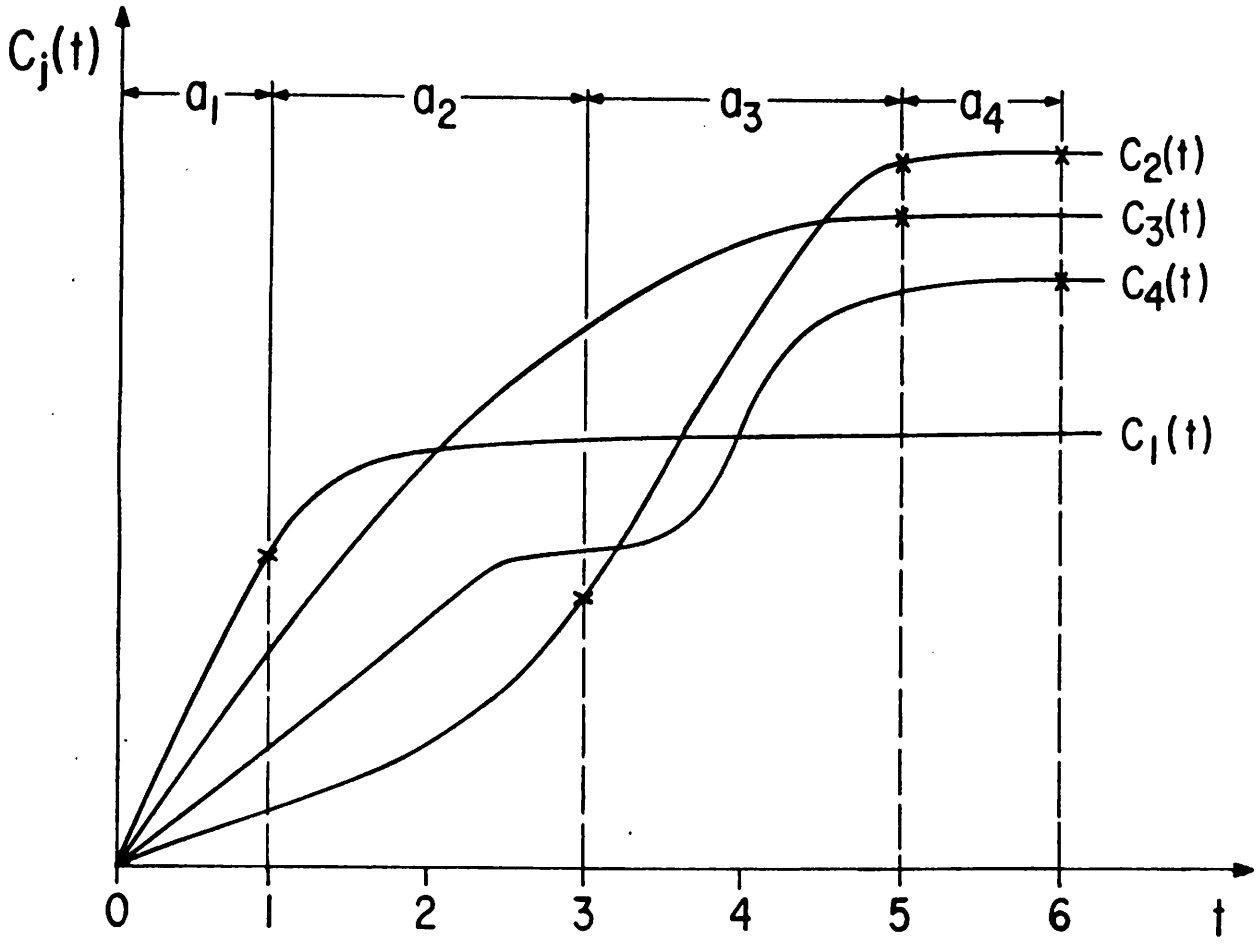

Fig. 2. Plot of Cost Functions for Example.

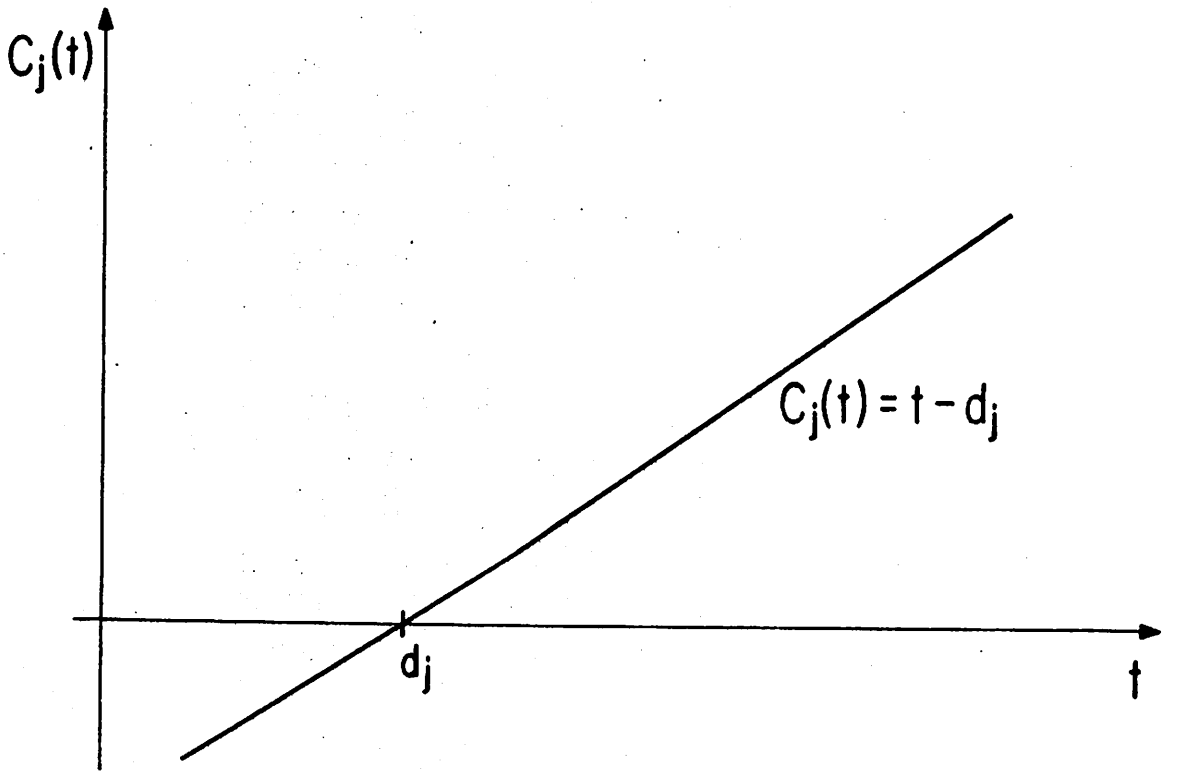

Fig. 3. Cost Function for Deadline Problem.