A "QUASI-POLYNOMIAL" ALGORITHM FOR SEQUENCING JOBS TO MINIMIZE TOTAL TARDINESS

by

Eugene L. Lawler

Memorandum No. ERL-M558

19 June 1975

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A "QUASI-POLYNOMIAL" ALGORITHM FOR SEQUENCING JOBS TO MINIMIZE TOTAL TARDINESS*

Eugene L. Lawler

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California at Berkeley

June 1975

Revised October 1975

Abstract

Suppose n jobs are to be processed by a single machine. Associated with each job j are a fixed integer processing time p_j, a deadline d_j, and a positive weight w_j. The weighted tardiness of job j in a given sequence is $w_j \max(0, C_j - d_j)$, where C_j is the completion time of job j. Assume that the weighting of jobs is "agreeable", in the sense that $p_i < p_j$ implies $w_i \geq w_j$. Under these conditions, it is shown that a sequence minimizing total weighted tardiness can be found by a dynamic programming algorithm with worst-case running time of $O(n^4P)$ or $O(n^5 p_{max})$, where $P = \sum p_j$ and $p_{max} = \max\{p_j\}$. The algorithm is "quasi-polynomial", since a true polynomial-bounded algorithm would be polynomial in $\log P$ or $\log p_{max}$.

Key words: sequencing, tardiness, algorithm, dynamic programming

---

*Research supported by National Science Foundation Grant GJ-43227X.

1. Introduction

Suppose n jobs are to be processed by a single machine. Associated with each job j are a fixed integer processing time p_j, a deadline d_j, and a positive weight w_j. The weighted tardiness of job j in a sequence is defined as $w_j \max\{0, C_j - d_j\}$, where C_j is the completion time of job j. The problem is to find a sequence which minimizes the total weighted tardiness, where the processing of the first job is to begin at time $t = 0$.

Let us assume that the weighting of jobs is agreeable, in the sense that $p_i < p_j$ implies $w_i \geq w_j$. Under these conditions, it is shown in this paper that an optimal sequence can be found by a dynamic programming algorithm with worst-case running time of $O(n^4P)$ or $O(n^5 p_{max})$, where $P = \sum p_j$, and $p_{max} = \max\{p_j\}$.

The proposed algorithm is distinguished from previous algorithms [4,6,12] for this problem in that its running time is bounded by a function that is polynomial, rather than exponential, in n. However, the present algorithm does not qualify as a true polynomial algorithm, since its running time is not bounded by a polynomial in the number of bits of data required to specify an instance of the problem [5]. To be polynomial in this sense, the running time bound should be polynomial in $\log P$ or $\log p_{max}$, rather than P or p_{max}.

It should be noted that the general weighted tardiness problem has been shown to be NP-complete [9]. This means that it is in the same equivalence class as the traveling salesman problem, the chromatic number problem, the three dimensional assignment problem, and other well-known hard problems, with respect to the existence of a polynomial algorithm. One may reasonably conjecture that there is no polynomial

algorithm for any of these problems.

The status of the agreeably-weighted tardiness problem is less clear. The proposed algorithm is, at best, "quasi-polynomial". On the other hand, none of the problem reductions known to the author can be used to establish that the problem is NP-complete.

It might be pointed out that the distinction between agreeble weightings and arbitrary weightings is clearer, in the case that one wishes to minimize the weighted number of tardy jobs. The agreeably weighted problem can be solved in polynomial time (specifically, $O(n \log n)$ [8]), whereas the general problem is NP-complete.

2. Theoretical Development

Theorem 1. Let the jobs have arbitrary weights. Let π be any sequence which is optimal with respect to the given deadlines $d_1,d_2,\ldots,d_n$, and let C_j be the completion time of job j for this sequence. Let d_j be chosen such that

$$\min(d_j,C_j) \leq d_j' \leq \max(d_j,C_j) .$$

Then any sequence π' which is optimal with respect to the deadlines $d_1',d_2',\ldots,d_n'$ is also optimal with respect to $d_1,d_2,\ldots,d_n$ (but not conversely).

Proof: Let T denote total weighted tardiness with respect to $d_1,d_2,\ldots,d_n$ and T' denote total weighted tardiness with respect to $d_1',d_2',\ldots,d_n'$. Let π' be any sequence which is optimal with respect to $d_1',d_2',\ldots,d_n'$, and let C_j' be the completion time of job j for this sequence. We have

$$T(\pi) = T'(\pi) + \sum_j A_j \ , \tag{1.1}$$

$$T(\pi') = T'(\pi') + \sum_j B_j \tag{1.2}$$

where, if $C_j \leq d_j$,

$$A_j = 0$$

$$B_j = -w_j \max(0,\min(C_j',d_j) - d_j') \ ,$$

and, if $C_j \geq d_j$,

$$A_j = w_j(d_j'-d_j)$$

$$B_j = w_j \max(0,\min(C_j',d_j') - d_j) \ .$$

Clearly $A_j \geq B_j$ and $\sum_j A_j \geq \sum_j B_j$. Moreover, $T'(\pi) \geq T'(\pi')$, because π' is assumed to minimize T'. Therefore the right hand side of (1.1) dominates the right hand side of (1.2). It follows that $T(\pi) \geq T(\pi')$ and π' is optimal with respect to $d_1,d_2,\ldots,d_n$. □

Theorem 2. Suppose the jobs are agreeably weighted. Then there exists an optimal sequence π in which job i precedes job j if $d_i \leq d_j$ and $p_i < p_j$, and in which all on time jobs are in nondecreasing deadline order.

Proof: Let π be an optimal sequence. Suppose i follows j in π, where $d_i \leq d_j$ and $p_i < p_j$. Then a simple interchange of i and j yields a sequence for which the total weighted tardiness is no greater. (Cf. proof of theorem in [10].) If i follows j, where $d_i \leq d_j$ and i and j are both on time, then moving j to the position immediately following i yields a sequence for which the total weighted tardiness is no greater. Repeated application of these two rules yields an optimal sequence satisfying the conditions of the theorem. □

In order to simplify exposition somewhat, let us assume for the purposes of the following theorem that all processing times are distinct. If processing times are not distinct, they may be perturbed infinitesimally without upsetting the assumption of agreeable weighting or otherwise changing the problem significantly. Hence there is no loss of generality.

Theorem 3. Suppose the jobs are agreeably weighted and numbered in nondecreasing deadline order, i.e. $d_1 \leq d_2 \leq \cdots \leq d_n$. Let job k be such that $p_k = \max_j \{p_j\}$. Then there is some integer δ, $0 \leq \delta \leq n-k$, such that there exists an optimal sequence π in which k is preceded by all jobs j such that $j \leq k+\delta$, and followed by all jobs j such that $j > k+\delta$.

Proof: Let C'_k be the latest possible completion time of job k in any sequence which is optimal with respect to deadlines $d_1, d_2, \ldots, d_n$. Let π be a sequence which is optimal with respect to the deadlines $d_1, d_2, \ldots, d_{k-1}, d'_k = \max(C'_k, d_k), d_{k+1}, \ldots, d_n$, and which satisfies the conditions of Theorem 2 with respect to these deadlines. Let C_k be the completion time of job k for π. By Theorem 1, π is optimal with respect to the original deadlines. Hence, by assumption, $C_k \leq d'_k$. Job k cannot be preceded in π by any job j such that $d_j > d'_k$, else job j would also be on time, in violation of the conditions of Theorem 2. And job k must be preceded by all jobs j such that $d_j \leq d'_k$. Let δ be chosen to be the largest integer such that $d_{k+\delta} \leq d'_k$ and the theorem is proved. □

3. Dynamic Programming Solution

Assume the jobs are agreeably weighted and numbered in nondecreasing deadline order. Suppose we wish to find an optimal sequence of jobs $1,2,\ldots,n$, with processing of the first job to begin at time t. Let k be the job with largest processing time. It follows from Theorem 3 that, for some δ, $0 \leq \delta \leq n-k$, there exists an optimal sequence in the form of:

(i) jobs $1,2,\ldots,k-1,k+1,\ldots,k+\delta$, in some sequence, starting at time t, followed by

(ii) job k, with completion time $C_k(\delta) = t + \sum_{j \leq k+\delta} p_j$, followed by,

(iii) jobs $k+\delta+1,k+\delta+2,\ldots,n$, in some sequence, starting at time $C_k(\delta)$.

By the well known principle of optimality it follows that the overall sequence is optimal only if the sequences for the subsets of jobs in (i) and (iii) are optimal, for starting times t and $C_k(\delta)$, respectively. This observation suggests a dynamic programming method of solution. For any given subset S of jobs and starting time t, there is a well-defined sequencing problem. An optimal solution for problem S, t can be found recursively from optimal solutions to problems of the form S', t', where S' is a proper subset of S and $t' \geq t$.

The subsets S which enter into the recursion are of a very restricted type. Each subset consists of jobs in an interval $i,i+1,\ldots,j$, with processing times strictly less than some value p_k. Accordingly, denote such a set by

$$S(i,j,k) = \{j' \mid i \leq j' \leq j,\ p_{j'} < p_k\} ,$$

and let

$T(S(i,j,k),t)$ = the total weighted tardiness for an optimal sequence of the jobs in $S(i,j,k)$, starting at time t.

By the application of Theorem 3 and the principle of optimality, we have:

$$T(S(i,j,k),t) = \min_{\delta}\{T(S(i,k+\delta,k'),t) + w_{k'}\max(0,C_{k'}(\delta)-d_{k'}) + T(S(k'+\delta+1,j,k'),C_{k'}(\delta)\} \qquad (3.1)$$

where k' is such that

$$p_{k'} = \max\{p_{j'} \mid j' \in S(i,j,k)\} ,$$

and

$$C_{k'}(\delta) = t + \sum p_{j'} ,$$

where the summation is taken over all jobs $j' \in S(i,k+\delta,k')$.

The initial conditions for the equations (3.1) are

$$T(\phi,t) = 0$$

$$T(\{j\},t) = w_j \max(0,t+p_j-d_j) .$$

It is easy to establish an upper bound on the worst-case running time required to compute an optimal sequence for the complete set of n jobs. There are no more than $O(n^3)$ subsets $S(i,j,k)$. (There are no more than n values for each of the indices, i, j, k. Moreover, several distinct choices of the indices may specify the same subset of jobs.) There are surely no more than $P = \sum p_j \leq np_{max}$ possible values of t. Hence there are no more than $O(n^3P)$ or $O(n^4p_{max})$ equations (3.1) to be solved. Each equation requires minimization over at most n alternatives and $O(n)$ running time. Therefore the overall running time is bounded by $O(n^4P)$ or $O(n^5p_{max})$.

At this point we have accomplished the primary objective of this paper, which is to present an algorithm which is polynomial in n. The remaining sections are devoted to a discussion of various computational refinements.

4. Refinements of the Algorithm

There are several possible refinements of the basic algorithm that may serve to reduce the running time significantly. However, none of these refinements is sufficient to reduce the theoretical worst-case complexity; some may actually worsen it.

Representation of Subsets

It should be noted that $S(i,j,k)$ may denote precisely the same subset of jobs as $S(i',j',k')$ even though $i \neq i'$, $j \neq j'$, $m \neq m'$. The notation used in (3.1) is employed only for convenience in specifying subsets. Obviously, the computation should not be allowed to be redundant.

State Generation

Only a very small fraction of the possible subproblems S,t are of significance in a typical calculation. Any practical scheme for implementing the recursion should have two phases. In the first, subproblem generation phase, one starts with the problem $S = \{1,2,\ldots,n\}$, $t = 0$ and successively breaks it down into only those subproblems S,t for which equations (3.1) need to be solved. In the second, recursion phase, one solves each of the subproblems generated in the first phase, working in the order opposite to that in which they were generated.

Restriction of δ

It is often not necessary for δ to range over all possible integer

values in (3.1). The range of δ can sometimes be considerably restricted by the technique described in the next section, thereby reducing the number of subproblems that need be generated and solved.

Shortcut Solutions

There are some "shortcut" methods of solution for the sequencing problem. Whenever one of these shortcut methods is applicable to a subproblem S,t generated in the first phase of the algorithm, it is unnecessary to solve that problem by recursion of the form (3.1) and no further subproblems need be generated from it. A discussion of shortcut solution methods is given in Section 6.

Branch-and-Bound

At least in the case of problems of moderate size, there appears to be relatively little duplication of the subproblems produced in the subproblem generation phase of the algorithm. In other words, the recursion tends to be carried out over a set of subproblems related by a tree structure, or something close to it. It follows that there may be some advantage to a branch-and-bound method, based on the structure of equations (3.1). Such a branch-and-bound method might have a very poor theoretical worst-case running time bound, depending on the nature of the bounding calculation and other details of implementation. However, if a depth-first exploration of the search tree is implemented. storage requirements could be very drastically reduced.

It is apparent that the form of recursion (3.1) furnishes a point of departure for the development of many variations of the basic computation.

5. Restriction of δ

The number of distinct values of δ over which minimization must be carried out in equations (3.1) can sometimes be reduced by appropriately invoking Theorems 1 and 2. If this is done in the state generation phase of the algorithm, there may be a considerable reduction in the number of subproblems which must be solved.

Consider a subproblem S, t. Let k be such that

$$p_k = \max_{j \in S} \{p_j\} ,$$

and assume that $p_k > p_j$, for all $j \in S - \{k\}$. We also assume that the jobs are numbered so that $d_1 \leq d_2 \leq \cdots \leq d_n$. The following algorithm determines distinct values δ_i, $i = 1,2,\ldots \leq n-k$, over which it is sufficient to carry out minimization in equations (3.1).

0. Set $i = 1$.

1. Set $d_k' = t + \sum_{j \in S'} p_j$, where $S' = \{j \mid d_j \leq d_k, j \in S\}$.

Comment: If job k has deadline d_k, then by Theorem 2 there exists an optimal sequence in which the completion time of job k is at least as large as d_k'.

2. If $d_k' > d_k$ set $d_k = d_k'$ and return to Step 1.

Comment: By Theorem 1, there exists a sequence which is optimal with respect to d_k' which is optimal with respect to d_k.

Let j be the largest index in S such that $d_j \leq d_k$. Set $\delta_i = j-k$.

Let $S'' = \{j \mid d_j > d_k, j \in S\}$. If S'' is empty, stop. Otherwise, let

j' be such that

$$d_{j'} = \min_{j \in S''} \{d_j\} ,$$

and set $d_k = d_{j'}$. Set $i = i+1$ and return to Step 1.

As an example of the application of the above procedure, consider the first test problem given in Appendix A of [1]. All $w_j = 1$. The p_j and d_j values are as follows:

j	1	2	3	4	5	6	7	8
p_j	121	79	147	83	130	102	96	88
d_j	260	266	269	336	337	400	683	719

Note that $k = 3$. Equation (3.1) yields:

$$T(\{1,2,\ldots,8\},0) = \min \left\{ \begin{array}{l} T(S(1,3,3),0) + 78 + T(S(4,8,3),347), \\ T(S(1,4,3),0) + 161 + T(S(5,8,3),430), \\ T(S(1,5,3),0) + 291 + T(S(6,8,3),560), \\ T(S(1,6,3),0) + 393 + T(S(7,8,3),662), \\ T(S(1,7,3),0) + 489 + T(S(8,8,3),758), \\ T(S(1,8,3),0) + 577 + T(\phi,846) \end{array} \right\}$$

Applying the procedure above, we obtain $\delta_1 = 3$, $\delta_2 = 5$ and the simplified equation:

$$T(\{1,2,\ldots,8\},0) = \min \left\{ \begin{array}{l} T(S(1,6,3),0) + 393 + T(S(7,8,3),662), \\ T(S(1,8,3),0) + 577 + T(\phi,846) \end{array} \right\} \quad (5.1)$$

6. Shortcut Solutions

"Shortcut" solutions are sometimes provided by generalizations of two well-known theorems for the unweighted tardiness problem [3].

Theorem 4. Let the jobs be given arbitrary weights. Let π be a sequence in which jobs are ordered in nonincreasing order of the ratios w_j/p_j. If all jobs are tardy, then π is optimal.

Proof: Note that

$$\sum w_j T_j = \sum w_j C_j + \sum w_j \max(0, d_j - C_j) \quad .$$

It is well-known [11] that π minimizes $\sum w_j C_j$. If all jobs are tardy, then each term in the second summation is zero and that sum is also minimized. □

Note that if jobs are agreeably weighted and processing times are distinct, then w_j/p_j-ratio order is equivalent to shortest processing time order.

Theorem 5. Let the jobs be given arbitrary weights. Let π be a sequence for which

$$\max_j \{w_j T_j\}$$

is minimum. If at most one job is tardy, then π is optimal.

Proof: Obvious. □

Note that in the unweighted case, nondecreasing deadline order minimizes maximum tardiness. In the case of arbitrary weightings, a minmax optimal order can be constructed by the $O(n^2)$ algorithm given in [7].

The application of these two theorems can be strengthened considerably by applying them to an earlier or a later set of deadlines induced by Theorems 1 and 2.

For a problem S, t let the jobs in S be numbered so that $p_1 > p_2 > \cdots > p_n$. New (earlier) deadlines d'_j for the application of Theorem 4 can be induced by the following algorithm.

0. Set $k = n$.

1. If $k = 1$, stop. Otherwise, set $k = k-1$.

2. Set $d'_k = d_k$.

3. Let $S^{(k)} = \{j \mid j \in S,\ d_j \geq d'_k,\ p_j > p_k\}$. Set $C_k = t + \sum_{j \in S - S^{(k)}} p_j$.

Comment: $S^{(k)}$ contains all those jobs which can be assumed to follow k by Theorem 2.

4. If $C_k < d'_k$, set $d'_k = C_k$ and return to Step 3. Otherwise, return to Step 1.

New (later) deadlines d'_k can be induced by the following algorithm:

0. Set $k = 1$.

1. If $k = n$, stop. Otherwise, set $k = k+1$.

2. Set $d'_k = d_k$.

3. Let $S^{(k)} = \{j \mid j \in S,\ d_j \leq d'_k,\ p_j < p_k\}$. Set $C_k = t + p_k + \sum_{j \in S^{(k)}} p_j$.

4. If $C_k > d'_k$, set $d'_k = C_k$ and return to Step 3. Otherwise, return to Step 1.

By Theorem 1, an optimal solution to the sequencing problem with respect to induced deadlines d_j', $j = 1,2,\ldots,n$, is optimal with respect to the deadlines d_j. Hence Theorems 4 and 5 can be applied with respect to the induced deadlines.

As an application of Theorems 4 and 5, let us solve equation (5.1). Consider first the application of Theorem 5 to $S(1,8,3)$, $t = 0$. If the jobs in $S(1,8,3)$ are sequenced in increasing d_j-order, i.e. 1, 2, 4, 5, 6, 7, 8, then jobs 5 and 6 are tardy so Theorem 5 does not apply. However, if induced deadlines are computed, it is found that $d_5' = 515$, with $d_j' = d_j$, for $j \neq 5$. When the jobs are sequenced in increasing d_j'-order, i.e. 1, 2, 4, 6, 5, 7, 8, no jobs are tardy with respect to d_j'. By Theorem 1, the sequence is optimal with respect to the original deadlines and $T(S(1,8,3),0) = 178$. Also by Theorem 5, $T(S(1,6,3),0) = 178$. And by Theorem 4, $T(S(7,8,3),662) = 194$. Hence (5.1) becomes:

$$T(\{1,2,\ldots,8\},0) = \min \begin{Bmatrix} 178+393+194, \\ 178+577+0 \end{Bmatrix}$$

$$= 755 ,$$

as indicated by Baker [1]. An optimal sequence is: 1, 2, 4, 6, 5, 7, 8, 3. Most of the test problems on the same list can be resolved with similar simplicity.

It should be mentioned that even in the case that Theorems 4 and 5 do not yield shortcut solutions, it may be possible to reduce the size of a subproblem with the following observation.

Theorem 6. Let k be such that $d'_k = \max\{d'_j \mid j \in S\}$, where the d'_j are induced deadlines obtained as above. Let P be the sum of the processing times of jobs in S. If $P+t \leq d'_k$, then

$$T(S,t) = T(S-\{k\},t) + w_k \max\{0,P+t-d_k\} .$$

Proof: Cf. [2]. □

7. Possibilities for a Polynomial Algorithm

As we have commented, the status of the agreeably weighted tardiness problem is unclear. The proposed algorithm is only "quasi-polynomial". However, no problem reduction has been devised to show that the problem is NP-complete, and one may still reasonably suppose that a polynomial algorithm does exist.

There are some possibilities that do not seem rewarding in searching for a polynomial algorithm. For a given set S, $T(S,t)$ is a piecewise linear function of t. If $T(S,t)$ were also convex, and all $w_j = 1$, then $T(S,t)$ could be characterized by at most $n+1$ linear segments, with successive slopes $0,1,2,\ldots,n$. The function $T(S,t)$ could then be computed in polynomial time, using equations (3.1). Unfortunately, $T(S,t)$ is not convex, as can be shown by simple counterexamples.

If the values of δ for which the minimum is obtained in (3.1) were monotonically nondecreasing with t, then this would also suggest a polynomial bounded algorithm. Unfortunately, there are simple counterexamples for this property, as well.

References

[1] K.R. Baker, Introduction to Sequencing and Scheduling, John Wiley, New York, 1974.

[2] S. Elmahgraby, "The One-Machine Sequencing Problem with Delay Costs," Journal of Industrial Engineering 19 (1974), 187-199.

[3] H. Emmons, "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness," Operations Research 17 (1969).

[4] M. Held and R.M. Karp, "A Dynamic Programming Approach to Sequencing Problems," J. Soc. Industr. Appl. Math. 10 (1962), 196-210.

[5] R.M. Karp, "On the Computational Complexity of Combinatorial Problems," Networks 5 (1975), 45-68.

[6] E.L. Lawler, "Sequencing Problems with Deferral Costs," Management Science 11 (1964), 280-288.

[7] E.L. Lawler, "Optimal Sequencing of a Single Processor Subject to Precedence Constraints," Management Science 19 (1973), 544-546.

[8] E.L. Lawler, "Sequencing to Minimize the Weighted Number of Tardy Jobs," to be published.

[9] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, "Complexity of Machine Scheduling Problems," to appear in Annals of Discrete Mathematics 1 (1976).

[10] A.H.G. Rinnooy Kan, B.J. Lageweg, J.K. Lenstra, "Minimizing Total Costs in One-Machine Scheduling," Operations Research (to appear).

[11] W.E. Smith, "Various Optimizers for Single-Stage Production," Naval Research Logistics Quarterly 3 (1956), 59-66.

[12] V. Srinivasan, "A Hybrid Algorithm for the One-Machine Sequencing Problem to Minimize Total Tardiness," Naval Research Logistics Quarterly 18 (1971).