[28] R.E. Gomory, An algorithm for integer solutions to linear programs, in: Graves and Wolfe, eds., *Recent Advances in Mathematical Programming* (McGraw-Hill, New York, 1963) pp. 269–302.
[29] R.E. Gomory, An algorithm for the mixed integer problem, RM–2597, RAND Corporation, 1960.
[30] G.A. Gorry and J.F. Shapiro, An adaptive group theoretic algorithm for integer programming problems, *Management Sci.* 17 (1971) 285–306.
[31] P.L. Hammer and S. Rudeanu, Pseudo-boolean programming, *Operations Res.* 17 (1969) 233–264.
[32] Hoang Tuy, Concave programming under linear constraints, in Russian; *Doklady Academii Nauk SSR* (1964) English translation in *Soviet Math.* (1964) 1437–1440.
[33] T.C. Hu, *Integer Programming and Network Flows* (Addison-Wesley, 1969) 432 + pp.
[34] R. Jeroslow, The principles of cutting-plane theory: Part I, Carnegie-Mellon University, February 1974.
[35] R. Jeroslow, The principles of cutting-plane theory, Part II: Algebraic methods, disjunctive methods, Man. Sci. Res. Rep. no. 370 (revised), Carnegie-Mellon University, September 1975.
[36] R. Jeroslow, Cutting-plane theory: Algebraic methods, February 1976.
[37] R. Jeroslow, Cutting-planes for relaxations of integer programs, Man. Sci. Res. Rep. No. 347, Carnegie-Mellon University, July 1974.
[38] R. Jeroslow, A generalization of a theorem of Chvátal and Gomory, in [41].
[39] E.L. Johnson, Integer programs with continuous variables, July 1974.
[40] E.L. Johnson, The group problem for mixed integer programming, *Math. Programming Study* 2, (1974), 137–179.
[41] O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, *Nonlinear Programming* 2 (Academic Press, New York, 1975).
[42] H. Minkowski, *Theorie der Konvexen Körper, insbesondere Begrundung ihres Oberflachenbegriffs*, Gesammelte Abhandlungen II, Leipzig, 1911.
[43] G. Owen, Cutting-planes for programs with disjunctive constraints, *J. Optimization Theory and Its Appl.* 11 (1973) 49–55.
[44] D. Prawitz, *Natural Deduction: A Proof-Theoretical Study*, Stockholm Studies in Philosophy 3 (Almqvist and Wiksell, Stockholm, 1965) 105 + pp.
[45] R.T. Rockafeller, *Convex Analysis* (Princeton University Press, Princeton, NJ, 1970).
[46] F.J. Shapiro, Generalized Lagrange multipliers in integer programming, *Operations Res.* 19 (1971) 68–76.
[47] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions I* (Springer, Berlin, 1970).
[48] H. Uzawa, A theorem on convex polyhedral cones, in: Arrow, Hurwicz, Uzawa, eds., *Studies in Linear and Nonlinear Programming* (Stanford University Press, Stanford, CA, 1958).
[49] H. Weyl, "Elementare Theorie der Konvexen Polyeder," *Comentarii Mathematici Helvetici* 7 (1935), pp. 290–306, English translation in *Contributions to the Theory of Games*, Kuhn and Tucker, eds., *Ann. Math. Studies* no. 24, Princeton, 1950.
[50] R.D. Young, Hypercylindrically-deduced cuts in zero-one integer programs, *Operations Res.* 19 (1971) 1393–1405.
[51] P. Zwart, Intersection cuts for separable programming, Washington University, St. Louis, January 1972.

Annals of Discrete Mathematics 1 (1977) 331–342
© North-Holland Publishing Company

A "PSEUDOPOLYNOMIAL" ALGORITHM FOR SEQUENCING JOBS TO MINIMIZE TOTAL TARDINESS*

Eugene L. LAWLER
Computer Science Division, University of California, Berkeley, CA

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 due date 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 $O(n^5p_{\max})$, where $P = \sum p_j$ and $p_{\max} = \max\{p_j\}$. The algorithm is "pseudopolynomial", since a true polynomial-bounded algorithm should be polynomial in $\sum \log_2 p_j$.

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 due date d_j, and a positive weight w_j. The tardiness of job j in a sequence is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job j. The problem is to find a sequence which minimizes total weighted tardiness, $\sum w_jT_j$, 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 $O(n^5p_{\max})$, where $P = \sum p_j$, and $p_{\max} = \max\{p_j\}$.

The proposed algorithm is distinguished from previous algorithms [5, 7, 15] 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 polynomial algorithm in the accepted sense of the term. This is because the running time is not bounded by a polynomial in the number of bits required to specify an instance of the problem in binary encoding. To be polynomial in this sense, the running time should be polynomial in $\sum \log_2 p_{ij}$, rather than P or $p_{\max}$.

Although the proposed algorithm is not polynomial with respect to binary encoding of data, it is polynomial with respect to an encoding in which the p_j values are expressed in unary notation. For this reason, we say that the algorithm is *pseudopolynomial*.

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

If the weights of jobs are unrestricted ("disagreeable"), then the weighted tardiness problem is NP-complete, even if all data are encoded in unary notation. (See proof in appendix.) This means that the existence of a pseudopolynomial algorithm is very unlikely. Or more precisely, such an algorithm exists if and only if there are similar algorithms for the traveling salesman problem, the three dimensional assignment problem, the chromatic number problem, and other well-known "hard" problems [6].

It should be mentioned that there is as yet no proof that the agreeably weighted tardiness problem is NP-complete with respect to binary encoding. Hence one may still hope to find a polynomial algorithm. Some unsuccessful attempts are described in the final section of this paper.

There are many closely related types of sequencing problems in which the distinctions between agreeable weighting and unrestricted weighting and between binary encoding and unary encoding are significant. For example, suppose all jobs have the same due date. Then the unrestricted weighted tardiness problem can be solved by a pseudopolynomial algorithm with $O(n^2P)$ complexity [10], whereas the agreeably weighted case yields to an $O(n \log n)$ procedure (SPT order). Or suppose we seek to minimize the weighted *number* of tardy jobs (with respect to arbitrary due dates). The unrestricted problem is NP-complete with respect to binary encoding, but can be solved in $O(nP)$ time [10]. The agreeably weighted case can be solved in $O(n \log n)$ time [9, 11].

2. Theoretical development

Theorem 1. *Let the jobs have arbitrary weights. Let π be any sequence which is optimal with respect to the given due dates $d_1, d_2, \ldots, d_n$, 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 due dates $d'_1, d'_2, \ldots, d'_n$ 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 $\Sigma_j A_j \geq \Sigma_j B_j$. 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. [13, proof of Theorem 1].) 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 applications 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 due date 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 due dates $d_1, d_2, \ldots, d_n$. Let π be a sequence which is optimal with respect to the due dates $d_1, d_2, \ldots, d_{k-1}, d'_k = \max(C'_k, d_k), d_{k+1}, \ldots, d_n$, and which satisfies the conditions of Theorem 2 with respect to these due dates. Let C_k be the completion time of job k for π. By Theorem 1, π is optimal with respect to the original due dates. Hence, by assumption, $C_k \leqslant d'_k$. Job k cannot be preceded in π by any job j such that $d_j > d'$, 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 $d_j \leqslant d'_k$. Let δ be chosen to be the largest integer such that $d_{k+\delta} \leqslant d'_k$ 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, $0 \leqslant \delta \leqslant n-k$, 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 $C_k(\delta) = t + \sum_{j \leqslant k+\delta} p_j$, 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 $t' \geqslant t$.

The subset 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

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

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:

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

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 $P = \sum p_j \leqslant np_{\max}$ possible values of t. Hence there are no more than $O(n^3 P)$ or $O(n^4 P_{\max})$ 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 $O(n^4 P)$ or $O(n^5 p_{\max})$.

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 $i \neq i'$, $j \neq j'$, $k \neq k'$. 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 equation (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 equation (3.1).

(0) Set $i = 1$.
(1) Set $d'_k = t + \Sigma_{j\in S'} p_j$, where $S' = \{j \mid d_j \leq d_k, j \in S\}$.

Comment. If job k has due date 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:

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

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

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

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

$$\sum w_jT_j = \sum w_jC_j + \sum w_j \max(0, d_j - C_j) - \sum w_jd_j.$$

It is well-known [14] that π minimizes $\sum w_jC_j$. 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*

$$\max_j \{w_jT_j\}$$

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

Proof. Obvious. □

Note that in the unweighted case, nondecreasing due date 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 [8].

The application of these two theorems can be strengthened considerably by applying them to an earlier or a later set of due dates 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 $k = n+1$.
(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) due dates 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 due dates d'_j, $j = 1, 2, \ldots, n$, is optimal with respect to the due dates d_j. Hence Theorems 4 and 5 can be applied with respect to the induced due dates.

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 due dates 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 due dates 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 "pseudopolynomial". 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 equation (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.

Appendix

The following proof of the unary NP-completeness of the weighted tardiness problem was communicated to the author by M.R. Garey and D.S. Johnson. An alternative proof has been developed by J.K. Lenstra. [12].

The so-called 3-partition problem was shown to be unary NP-complete in [4]. This problem is as follows. Given a set of $3n$ integers $a_1, a_2, \ldots, a_{3n}$ between 1 and $B-1$ such that $\sum a_i = nB$, we wish to determine whether there is a partition of the a_i's into n groups of 3, each summing exactly to B.

The corresponding scheduling problem:

"X"-jobs:	$X_i, 1 \leq i \leq n.$
"A"-jobs:	$A_i, 1 \leq i \leq 3n.$
Processing times:	$p(X_i) = L = (16B^2)\dfrac{n(n+1)}{2} + 1, 1 \leq i \leq n,$
	$p(A_i) = B + a_i, 1 \leq i \leq 3n.$
Weights:	$w(X_i) = W = (L + 4B)(4B)\dfrac{n(n+1)}{2} + 1, 1 \leq i \leq n,$
	$w(A_i) = p(A_i) = B + a_i, 1 \leq i \leq 3n.$
Due dates:	$d(X_i) = iL + (i-1)4B, 1 \leq i \leq n,$
	$d(A_i) = 0, 1 \leq i \leq 3n.$

Question: Is there a schedule π with total weighted tardiness $T(\pi) \leq W - 1$?

Suppose the desired partition exists. We may assume without loss of generality that the groups are $\langle a_{3j-2}, a_{3j-1}, a_{3j}\rangle$, $1 \leq j \leq n$. Consider the following ordering of the jobs:

$$\pi = \langle X_1, A_1, A_2, A_3, X_2, A_4, A_5, A_6, X_3, \ldots, X_i,$$
$$A_{3i-2}, A_{3i-1}, A_{3i}, \ldots, X_n, A_{3n-2}, A_{3n-1}, A_{3n}\rangle.$$

By assumption $\sum_{i=-2}^{0} p(A_{3j-i}) = 4B$ for $1 \leq j \leq n$. Thus X_i will finish at time $iL + (i-1)4B = d(X_i)$, $1 \leq i \leq n$, and so none of the X-jobs are tardy. On the other hand, *all* the A jobs are tardy, with tardinesses equal to their completion

times. For each j, $1 \leq j \leq n$, the three jobs A_{3j-2}, A_{3j-1}, and A_{3j} all finish by $j(L + 4B)$, and their total weight is $4B$. Hence their collective weighted tardiness is at most $j(L + 4B)4B$. Hence

$$T(\pi) \leq \sum_{j=1}^{n} j(4B)(L + 4B) = \frac{n(n+1)}{2}(4B)(L + 4B) = W - 1,$$

and π is the desired schedule.

Conversely, suppose that π is such that $T(\pi) \leq W - 1$. Clearly no X-job can be tardy, for even a tardiness of 1 would yield $T(\pi) \geq W$. Now define W_i to be the total weight of the A-jobs which follow i X-jobs, with $W_{n+1} = 0$ by convention. Then

$$T(\pi) \geq \sum_{i=1}^{n} (W_i - W_{i+1})(iL) = L \sum_{i=1}^{n} W_i.$$

Since all X-jobs meet their due date, we must have $W_i \geq (n - i + 1)4B$, $1 \leq i \leq n$. Suppose some $W_i \geq (n - i + 1)4B + 1$. Then

$$\sum W_i \geq 1 + \sum_{i=1}^{n} (n - i + 1)4B = \frac{n(n+1)}{2}(4B) + 1.$$

This would imply that

$$T(\pi) \geq L(4B)\left(\frac{n(n+1)}{2}\right) + 16B^2\frac{n(n+1)}{2} + 1 = W,$$

a contradiction.

Thus $W_i = (n - i + 1)4B$, $1 \leq i \leq n$. From this we conclude that the set of A-jobs between X_i and X_{i+1} in π has total weight $4B$, $1 \leq i \leq n - 1$, and similarly for the set of A-jobs following X_n. Since all A-jobs have $B + 1 \leq w(A) \leq 2B - 1$, each such set must contain exactly 3 jobs. These n groups of 3 jobs correspond to the desired partition. □

Acknowledgement

The author wishes to acknowledge the helpful comments of Dr. Alexander Rinnooy Kan.

References

[1] K.R. Baker, Introduction to Sequencing and Scheduling, (Wiley, New York, 1974).
[2] S. Elmahgraby, The one-machine sequencing problem with delay costs, J. Industrial Eng. 19 (1974) 187–199.
[3] H. Emmons, One-machine sequencing to minimize certain functions of job tardiness, Operations Res. 17 (1969) 701–715.

[4] M.R. Garey and D.S. Johnson, Complexity results for multiprocessor scheduling under resource constraints, SIAM J. Comp. 4 (1975) 397–411.
[5] M. Held and R.M. Karp, A dynamic programming approach to sequencing problems, J. Soc. Industr. Appl. Math. 10 (1962) 196–210.
[6] R.M. Karp, On the computational complexity of combinatorial problems, Networks 5 (1975) 45–68.
[7] E.L. Lawler, Sequencing problems with deferral costs, Management Sc. 11 (1964) 280–288.
[8] E.L. Lawler, Optimal sequencing of a single processor subject to precedence constraints, Management Sc. 19 (1973) 544–546.
[9] E.L. Lawler, Sequencing to minimize the weighted number of tardy jobs, to appear in Rev. Française Automat. Informat. Recherche Opérationnelle, Suppl. to 10 (1976) 27–33.
[10] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, Management Sc. 16 (1969) 77–84.
[11] J.M. Moore, An n job, one machine scheduling algorithm for minimizing the number of late jobs, Management Sci. 1 (1968) 102–109.
[12] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math. 1 (1977) 343–362.
[13] A.H.G. Rinnooy Kan, B.J. Lageweg, J.K. Lenstra, Minimizing total costs in one-machine scheduling, Operations Res. 23 (1975) 908–927.
[14] W.E. Smith, Various optimizers for single-stage production, Naval Res. Logistics Quarterly 3 (1956) 59–66.
[16] V. Srinivasan, A hybrid algorithm for the one-machine sequencing problem to minimize total tardiness, Naval Res. Logistics Quarterly 18 (1971) 317–327.

Annals of Discrete Mathematics 1 (1977) 343–362
© North-Holland Publishing Company

COMPLEXITY OF MACHINE SCHEDULING PROBLEMS

J.K. LENSTRA
Mathematisch Centrum, Amsterdam, The Netherlands

A.H.G. RINNOOY KAN
Erasmus University, Rotterdam, The Netherlands

P. BRUCKER
Universität Oldenburg, G.F.R.

We survey and extend the results on the complexity of machine scheduling problems. After a brief review of the central concept of NP-completeness we give a classification of scheduling problems on single, different and identical machines and study the influence of various parameters on their complexity. The problems for which a polynomial-bounded algorithm is available are listed and NP-completeness is established for a large number of other machine scheduling problems. We finally discuss some questions that remain unanswered.

1. Introduction

In this paper we study the complexity of machine scheduling problems. Section 2 contains a brief review of recent relevant developments in the theory of computational complexity, centering around the concept of NP-completeness. A classification of machine scheduling problems is given in Section 3. In Section 4 we present the results on the complexity of these problems: a large number of them turns out to be NP-complete. Quite often a minor change in some parameter transforms an NP-complete problem into one for which a polynomial-bounded algorithm is available. Thus, we have obtained a reasonable insight into the location of the borderline between "easy" and "hard" machine scheduling problems, although some questions remain open. They are briefly discussed in Section 5.

2. Complexity theory

Recent developments in the theory of computational complexity as applied to combinatorial problems have aroused the interest of many researchers. The main credit for this must go to S.A. Cook [7] and R.M. Karp [25], who first explored the relation between the classes $\mathcal{P}$ and $\mathcal{NP}$ of (language recognition) problems solvable by *deterministic* and *non-deterministic* Turing machines respectively, in a number of steps *bounded by a polynomial in the length of the input.* With respect to