STRONGLY POLYNOMIAL ALGORITHMS FOR THE HIGH MULTIPLICITY SCHEDULING PROBLEM

DORIT S. HOCHBAUM
University of California, Berkeley, California

RON SHAMIR
Rutgers University, Piscataway, New Jersey

(Received April 1988; revisions received March 1989, January 1990; accepted January 1990)

A high multiplicity scheduling problem consists of many jobs which can be partitioned into relatively few groups, where all the jobs within each group are identical. Polynomial, and even strongly polynomial, algorithms for the standard scheduling problem, in which all jobs are assumed to be distinct, become exponential for the corresponding high multiplicity problem. In this paper, we study various high multiplicity problems of scheduling unit-time jobs on a single machine. We provide strongly polynomial algorithms for constructing optimal schedules with respect to several measures of efficiency (completion time, lateness, tardiness, the number of tardy jobs and their weighted counterparts). The algorithms require a number of operations that are polynomial in the number of groups rather than in the total number of jobs. As a by-product, we identify a new family of nxn transportation problems which are solvable in $O(n \log n)$ time by a simple greedy algorithm.

A scheduling problem is typically stated as follows: There is a list of jobs, and each job has a set of characteristic parameters (due date, weight, etc.). The goal is to find a schedule that minimizes a specified measure of efficiency (lateness, tardiness, etc.). We shall call this the *classical problem* representation. Here we consider what we call *high multiplicity* problems, in which the jobs can be partitioned into relatively few groups (or *types*), and in each group all the jobs are identical, i.e., they have the same set of parameters. The number of jobs of a specific type is called the *multiplicity* of that type. This paper studies various scheduling problems on a single machine with unit-time, high multiplicity jobs.

For a high multiplicity (henceforth abbreviated HM) problem, denote the number of distinct types by n, and the total number of unit jobs by P. A classical input representation, listing the parameters for each job, will be polynomial in P. A more compact representation of the input lists the parameters for each type *once*, along with the multiplicity of that type. The length of this input representation depends only *logarithmically* on P. Consequently, a polynomial algorithm for a classical problem, whose complexity is polynomial in the number of jobs P, becomes exponential for HM problems. Even a strongly polynomial algorithm for a classical problem is merely guaranteed to by *pseudopolynomial* for the corresponding HM problem.

Our aim is to find polynomial and, if possible, *strongly polynomial* algorithms for HM problems. As a by-product, we also identify a new class of transportation problems, which are solvable by a greedy algorithm, in time complexity which is much faster than that of any general transportation algorithm.

The idea of exploiting high multiplicity in combinatorial optimization is not new, although we believe this paper is its first general, explicit exposition. It has implicitly been discussed in scheduling (Psaraftis 1980), for the traveling salesperson problem (Rothkopf 1966, Cosmadakis and Papadimitriou 1984), and for graphs (Iwano and Steiglitz 1987, Lengauer 1982, 1986, 1987). Note also that the classical transportation problem is in fact an HM version of the assignment problem and that the cutting stock problem is an HM bin packing problem. For a discussion of applications of HM modeling to both polynomially solvable and NP-hard problems, see Hochbaum and Shamir (1988).

HM scheduling problems may also be interpreted as classical scheduling problems, with a different kind of objective: Each type is considered as a superjob of length equal to its multiplicity, and the goal is to find a *preemptive* scheduling of the superjobs. The objective function takes into account the contribution of each unit (rather than each superjob) to the total cost. Allowing preemption introduces a new difficulty: The mere description of a preemptive solution may require space proportional to P, in which case any algorithm producing such a solution will be exponential. Hence, for

Subject classifications: Computers/computer science: strongly polynomial algorithms. Networks/graphs, flow algorithms: greedy algorithms for transportation. Production/scheduling: sequencing, deterministic, single machine.

0030-364X/91/3904-0648 $01.25

problems with preemptive solutions, it is essential to prove that there exists an optimal solution whose description length is polynomial in n.

1. THE WEIGHTED NUMBER OF TARDY JOBS PROBLEM

The problem is defined as follows: There are n types of jobs, and there are p_i identical unit-time jobs of type i; p_i is called the *multiplicity* of type i. Type i jobs have due date d_i and weight w_i. The total number of jobs is $P = \sum_i p_i$. All the numbers that define the problem are integers. A *schedule* is an assignment of the P jobs to the distinct integers $1, \ldots, P$, such that type i jobs are assigned exactly p_i integers for $i = 1, \ldots, n$. The assignment of a job to number t is interpreted as scheduling that job to be processed in the time interval $(t-1, t]$. In that case, we shall say that this job is *processed at time* t. More formally, a schedule is a function σ: $\{1, \ldots, P\} \to \{1, \ldots, n\}$ and is feasible if $|\sigma^{-1}(i)| = p_i$ for $i = 1, \ldots, n$. In this section, we address the HM problem of finding a schedule that minimizes the weighted number of tardy jobs, i.e., minimizes $\sum_{t=1}^{P} I(t - d_{\sigma(t)}) w_{\sigma(t)}$ where I is the indicator function satisfying $I(x) = 1$ if $x > 0$ and $i(x) = 0$ otherwise.

In the unweighted case, a solution is easily obtained by the following procedure, which is a specialization of the classical algorithm of Moore (1968; see also French 1982): Schedule type after type, from $t = 0$ forward, according to an increasing order of due dates. If units have been assigned consecutively up to time t, and i is the next type to be scheduled, schedule $\min(p_i, d_i - t)$ units of type i, and postpone the remaining units of that type, if any, to the end of the schedule. Update the partial sum and repeat. This procedure obviously requires $O(n \log n)$ steps. Note that the optimal schedule may have to be preemptive. The extension of Moore's algorithm to the HM problem in which jobs have integer (nonunit, and not necessarily equal) processing times is also straightforward, by postponing a sufficient number of jobs with the largest processing times among those already scheduled.

For the nonunit job model, the weighted problem is NP-hard (Karp 1972). It is solvable by a pseudopolynomial algorithm due to Lawler and Moore (1969). For unit-time jobs, the assignment algorithm solves the problem in $O(P^3)$ time. Lawler's algorithm for the *agreeable weights* case (Lawler 1976) solves the problem in $O(P \log P)$ time. However, both algorithms are not polynomial for the HM problem. We will present an algorithm that solves this problem in strongly polynomial time.

Assume that the jobs are numbered so the $d_1 \leqslant d_2 \leqslant \cdots \leqslant d_n$. For notational simplicity we assume without loss of generality that all due dates are distinct. Define $d_0 := 0$, $d_{n+1} := P$. For $k = 1, \ldots, n+1$, let $\delta_k := d_k - d_{k-1}$, and denote by I_k the interval $(d_{k-1}, d_k]$. We formulate a transportation problem as follows: Let $X_{i,j}$ be the nubmer of units of type i processed in interval I_j, $i = 1, \ldots, n$, $j = 1, \ldots, n+1$. Define

$$C_{i,j} = \begin{cases} 0 & \text{if } d_j \leqslant d_i \\ w_i & \text{if } d_j > d_i. \end{cases} \tag{1}$$

The problem is thus equivalent to

$$\text{minimize} \sum_{i,j} C_{i,j} X_{i,j}$$

$$\text{subject to} \sum_j X_{i,j} = p_i \qquad i = 1, \ldots, n$$

$$\sum_i X_{i,j} = \delta_j \qquad j = 1, \ldots, n+1$$

$$X_{i,j} \geqslant 0 \quad \text{integer } i = 1, \ldots, n,\ j = 1, \ldots, n+1.$$

The resulting transportation problem can be solved by any of the new, strongly polynomial minimum cost network flow algorithms (see Orlin 1988 and the references thereof). In what follows, we use the special structure of our transportation problem to construct an algorithm with a much better running time. The result will hold for every transportation problem which has the cost structure (1). In fact, we can prove the result for a more general class of transportation problems, that is, for those with $n \times l$ cost matrix of the form

$$C_{i,j} = \begin{cases} 0 & \text{if } j \leqslant k_i \\ w_i & \text{if } j > k_i \end{cases} \tag{2}$$

where $i \leqslant k_i \leqslant l$, $i = 1, \ldots, n$. The algorithm is therefore presented for solving transportation problems, using the standard transportation notation. The algorithm used is the greedy type: It scans all the decision variables $X_{i,j}$ in a prescribed order, and successively maximizes each variable in turn. The scanning is done row after row, in decreasing order of w_i, and within each row by moving cyclically leftwards, beginning from the rightmost variable of cost zero in that row.

Algorithm A

Reorder rows such that $w_1 \geqslant w_2 \geqslant \ldots \geqslant w_n$.
For $i = 1, \ldots, n$ do
  For $j = 1, \ldots, l$ do
    let $t = (k_i - j) \bmod l + 1$
    $u \leftarrow \min(a_i, b_t)$
    $X_{i,t} \leftarrow u$
    $a_i \leftarrow a_i - u$
    $b_t \leftarrow b_t - u$
  repeat
repeat
end.

Theorem 1. *Algorithm A constructs an optimal solution to the transportation problem with cost matrix of the form* (2) *and can be implemented in* $O(m \log m)$ *time, where* $m = \max(n, l)$.

Proof. We first prove the correctness of the algorithm: Let $(\bar{Y}_{i,j})$ be a solution generated by Algorithm A, and assume that it is not optimal. Among all the optimal solutions, let $(\bar{X}_{i,j})$ be one for which *the first point of disagreement* with $\bar{Y}$ is *latest*, when compared according to the scanning order dictated by Algorithm A. Without loss of generality, let 1 be the first row on which $\bar{X}$ and $\bar{Y}$ differ, and j be the first column according to the scanning order on which they differ within that row. In this case $\bar{X}_{1,j} < \bar{Y}_{1,j}$ because Algorithm A maximizes every variable in turn. Let k be the first column according to the scanning order for which $\bar{X}_{1,k} > \bar{Y}_{1,k}$. (There must be such a column since $\sum_i \bar{X}_{1i} = \sum_i \bar{Y}_{1i} = a_1$.)

Without loss of generality, let 2 be a row of weight $w_2 \leqslant w_1$, satisfying $\bar{X}_{2,j} > 0$. (There must be such a row since $\bar{X}_{1,j} < \bar{Y}_{1,j}$, row 1 is the one on which the first disagreement occurs and the column totals are equal.) We now show that $\bar{X}_{1,j}$ can be increased and $\bar{X}_{1,k}$ can be decreased by the simple augmentation

$$\begin{aligned} \bar{X}_{1,j} &\leftarrow \bar{X}_{1,j} + \delta \qquad & \bar{X}_{1,k} &\leftarrow \bar{X}_{1,k} - \delta \\ \bar{X}_{2,j} &\leftarrow \bar{X}_{2,j} - \delta \qquad & \bar{X}_{2,k} &\leftarrow \bar{X}_{2,k} + \delta \end{aligned} \tag{3}$$

where $\delta := \min\{\bar{Y}_{1,j} - \bar{X}_{1,j}, \bar{X}_{1,k}, \bar{X}_{2,j}\}$.

The change in the solution value following the augmentation is $\Delta \cdot \delta$, where $\Delta = C_{1,j} - C_{1,k} + C_{2,k} - C_{2,j}$. There are three possible cases:

Case 1. $C_{1,j} < C_{1,k}$. In this case necessarily $C_{1,k} = w_1$ and $C_{1,j} = 0$ (recall that there are only two distinct values per row, and the smaller appears first in the scanning order dictated by the algorithm). On the other hand, $C_{2,k} - C_{2,j} \leqslant w_2 \leqslant w_1$. Therefore, $\Delta = -w_1 + (C_{2,k} - C_{2,j}) \leqslant -w_1 + w_2 \leqslant 0$.

Case 2. $C_{1,j} = C_{1,k}$. In this case $k < j$ (that is, column k appears to the left of column j in the matrix), since $\bar{X}_{1,j}$ was chosen as the *rightmost* column in row 1 according to the scanning order on which $\bar{X}$ and $\bar{Y}$ disagree. From the structure of the matrix it follows that $C_{2,k} \leqslant C_{2,j}$, and so: $\Delta = 0 + C_{2,k} - C_{2,j} \leqslant 0$.

Case 3. $C_{1,j} > C_{1,k}$. This cannot happen because we scan all the entries of lower cost first in each row, and j was scanned prior to k in row 1.

Repeating the process with the new $\bar{X}$ if necessary, We eventually get $\bar{X}_{1,j} = \bar{Y}_{1,j}$. The cost of the new $\bar{X}$ could only decrease along the process. But now $\bar{X}$ is an optimal solution for which the first point of disagreement with $\bar{Y}$ appears later along the scanning order, a contradiction.

To prove the complexity of the algorithm, note first that sorting the vectors w and d requires $O(m \log m)$ time, where $m = \max(n, l)$. In the solution $\bar{Y}$, at most $n + l - 1$ variables will be positive because whenever a variable attains a positive value either the supply at its source or the demand at its destination (or both) are exhausted. Hence, a description of the solution requires $O(m)$ space only. Indeed, it has been shown in Hochbaum and Shamir (1990) that a greedy algorithm which scans variables row after row and cyclically in each row can be implemented in $O(n + l)$ time. Hence, after sorting, the solution $\bar{Y}$ can be computed in $O(m)$ steps.

Note that the sorting of w and d is independent of the supply and demand quantities. If one has to solve several problems with the same costs, the sorting needs to be done only once, and the solution time of each subsequent problem will be linear.

Corollary 1. *Algorithm A can be used to construct an optimal solution for the problem of minimizing the weighted number of tardy jobs in* $O(n \log n)$ *time.*

An alternative proof of the above result, based on the notion of Monge sequences (Hoffman 1963), is given in Hochbaum and Shamir (1988). Note that the corollary also gives another proof to Lawler's result (Lawler 1976) that the classical problem with unit-time jobs is solvable in $O(P \log P)$ operations.

2. THE TOTAL WEIGHTED TARDINESS PROBLEM

The input for the total weighted tardiness problem is the same as in the previous section. Here we seek a schedule σ that minimizes $\sum_{t=1}^{P} \max(0, t - d_{\sigma(t)}) \cdot w_{\sigma(t)}$. The essential difference between the HM problem and the classical one is that here each job unit contributes to the total cost, where for the classical problem only the finish time of the last unit of each type contributes to the total cost.

The classical total tardiness problem is solvable in pseudopolynomial time (Lawler 1977), and was shown to be NP-hard (Du and Leung 1990). The classical weighted problem is strongly NP-hard (Lawler 1977), but when all the n jobs are of equal length, the problem can be solved in $O(n^3)$ steps by solving an assignment problem. For the HM problem this gives a pseudopolynomial solution in $O(P^3)$ steps. In this assignment problem, the cost of assigning a unit of type i to a time slot

grows linearly with the positive distance of that time slot from d_i, and thus is not fixed within each due date interval as it was in the previous problem. Therefore, the approach taken in Section 1 is inapplicable.

At this point, it is an open question whether this problem can be solved in strongly polynomial time. One prerequisite to the existence of a strongly polynomial algorithm is that the output can be described in strongly polynomial time. Such a description indeed exists for our problem: For each due date interval, as defined in the previous section, the optimal schedule is completely determined by the number of jobs of each type which are assigned to that interval. Ordering of the jobs within each interval is then done by first scheduling all the types that are tardy in that interval in decreasing order of weights, and then arbitrarily scheduling all the nontardy types, where jobs of the same type are grouped together. The description of that solution consists of $O(n^2)$ numbers of a length at most $\log P$ bits each. Hence, it is possible that a strongly polynomial algorithm exists for this problem.

On the other hand, all the optimal schedules may well be preemptive. We present an example of such a situation below. It implies that a mere sequencing of the job types may be suboptimal. The same example also rules out the possibility of a solution via reformulation as a transportation problem with the same partition into due date intervals as in Section 1: For such a problem there should be a basic, optimal solution in which the basic arcs form a tree. In the example, the positive flow arcs (and, hence, the basic arcs) contain a cycle.

Example. Let $w_1 = 10$, $d_1 = 2$, $p_1 = 2$, $w_2 = 20$, $d_2 = 4$, $p_2 = 4$, $w_3 = 11$, $d_3 = 4$, $p_3 = 1$. The only optimal solutions are the following orders of the unit jobs: 1, 2, 2, 2, 2, 3, 1 and 2, 1, 2, 2, 2, 3, 1. Since jobs of types 1 and 2 appear together in intervals (0, 2] and (4, 7], the optimal solutions do not correspond to any basic solution of the transportation problem.

There are two special cases for which we have strongly polynomial algorithms:

Case 1. Very agreeable weights. (This is paraphrased after Lawler's "agreeable weights" 1976). We call weights *very agreeable* if whenever $d_i \leqslant d_j$ also $w_i \geqslant w_j$. Here an optimal schedule is achieved in $O(n \log n)$ time by sequencing types in increasing order of due dates, and in decreasing order of weights among types of equal due date. The proof follows by an interchange argument. Note that problems in which all due dates are equal or all weights are equal are included in this case.

Case 2. The case of two types. In this case there are only two types of multiplicities p_1 and p_2 and weights w_1 and w_2, respectively. Note that even for this simple case enumeration is not polynomial: The schedule is completely determined by the number of type 1 jobs in two of the three due date intervals, but the number of such possibilities is already quadratic in p_1, and, hence, is not polynomial in $\log P$. This problem can be solved in strongly polynomial time, which is a constant in this case.

Since the case of equal due dates has already been discussed, assume without loss of generality that $d_1 < d_2$. Assume also that $w_1 < w_2$, since if $w_1 \geqslant w_2$ this is again case 1. Let $C_i(t) = \max[0, w_i(t - d_i)]$, $i = 1, 2$ be the tardiness functions of the two types. Consider the difference function $D(t) = C_1(t) - C_2(t)$. We regard $D(t)$ as a function of the real valued argument t. This function is piecewise-linear, and concave for $t \geqslant d_1$. The optimal scheduling will be performed according to this difference function: Assign the type 2 units to the p_2 unit intervals of the *highest value* of $D(t)$. A formal description follows.

The Two-Types Algorithm

Let b be the integer that satisfies $D(b) > 0$ and $D(b + 1) \leqslant 0$.

Step 1. If $P - b \geqslant p_1$, then assign all the jobs of type 1 to the interval $(P - p_1, P]$ and go to Step 4. Else, assign $P - b$ of the 1-jobs to the interval $(b, P]$ and set $p_1 \leftarrow p_1 - (P - b)$.

Step 2. If $p_1 \leqslant d_1$, assign the remaining 1-jobs to any positions in the interval $(0, d_1]$, and go to Step 4. Else, assign d_1 1-jobs to the interval $(0, d_1]$ and set $p_1 \leftarrow p_1 - d_1$.

Step 3. Let x be a value such that the total length of the two intervals described by the set $\{t \mid D(t) \leqslant x$ and $d_1 < t \leqslant b\} = (d_1, t_1] \cup (t_2, b]$ is equal to p_1. If t_1 is integer, then assign the remaining 1-jobs to the above pair of intervals. Otherwise, compare the total cost of the assignment of the 1-jobs to the intervals $(d_1, \lfloor t_1 \rfloor]$, $(\lfloor t_2 \rfloor, b]$ and the remaining 2-jobs to the $(\lfloor t_1 \rfloor, \lfloor t_2 \rfloor]$ interval, to the cost of assigning the 1-jobs to the intervals $(d_1, \lceil t_1 \rceil]$, $(\lceil t_2 \rceil, b]$ and the remaining 2-jobs to $(\lceil t_1 \rceil, \lceil t_2 \rceil]$. Assign the 1-jobs to the intervals of lower cost between the above.

Step 4. Assign the 2-jobs to the remaining free intervals.

3. SUMMARY AND DISCUSSION

We introduce in this paper the concept of high multiplicity, and illustrate its use for several single-machine scheduling problems. Our concept of the efficiency of

algorithms in this analysis ties up with the concept of strong polynomiality.

Table I summarizes the results for the high multiplicity model, and compares them to the results of the classical models, both for unit and nonunit time jobs. In addition to the results in Sections 1 and 2, we include in the table results for several other cost criteria. Some of these can be derived by relatively straightforward adaptations of algorithms for the corresponding classical problems. These results are explained in more detail in Hochbaum and Shamir (1988). The rightmost column in the table indicates whether the solution for the HM problem is preemptive or not. The upper line for each entry describes the type of algorithm which provides the solution. The complexity bound for that algorithm appears in the lower line. (Types of algorithms are indicated only for the HM problems, or when the distinction between the classical model and the HM model is of special interest.) EDD is the earliest due date rule, and WSPT is the weighted shortest processing time rule.

Since the completion of this work we were able to make progress in the general problem of total weighted tardiness by providing a special purpose algorithm that solves a quadratic integer transportation problem which models this problem (Hochbaum, Shamir and Shanthikumar 1988). The algorithm is polynomial, and the number of steps it requires is independent of the multiplicities and due dates.

The HM model reopens for investigation a wide variety of basic scheduling problems. These include problems of multimachine scheduling, with precedence constraints, and with job types that are not necessarily of unit length. The authors addressed the HM problem of minimizing the number of tardy job units under release time constraints, and devised $O(n \log n)$ and $O(n^2)$ algorithms for the unweighted and weighted problems, respectively (Hochbaum and Shamir 1990).

Issues related to the idea of high multiplicity come up in contexts other than scheduling. For instance, in the bin packing problem, the idea of creating a small number of piece sizes and clustering pieces of the same size in one set is useful in approximations (Hochbaum and Shmoys 1987). The relevant question is then whether one can solve such HM problems in time depending only on the number of piece sizes, rather than the total number of pieces. This question is still open.

ACKNOWLEDGMENT

The research of the first author was supported in part by the National Science Foundation under grant ECS-85-01988 and by Office of Naval Research contract no. N00014-88-K-0377. The research of the second author was supported in part by an Allon fellowship, by AFOSR grants 89-0512 and 90-0008, and by DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), a National Science Foundation Science and Technology Center under grant NSF-STC88-09648.

Table I
Summary of Results for the High Multiplicity Model

Minimization Criterion	Unit-Job Model	Nonunit-Job Model	High Multiplicity Model	Pre-emption
Total weighted completion time		WSPT	Decreasing weights	No
Total weighted lateness	$O(P \log P)$	$O(n \log n)$	$O(n \log n)$	
Max. weighted completion time	$O(P \log P)$	$O(n \log n)$	Decreasing weights $O(n \log n)$	No
Max. tardiness			EDD	No
Max. lateness	$O(P \log P)$	$O(n \log n)$	$O(n \log n)$	
Max. weighted tardiness			Greedy backwards	No
Max. weighted lateness	$O(P \log^2 P)$	$O(n \log^2 n)$	$O(n \log^2 n)$	
No. of tardy jobs	$O(P \log P)$	$O(n \log n)$	EDD + rejections $O(n \log n)$	Yes
Weighted no. of tardy jobs	$O(P \log P)$	Pseudopolynomial NP-hard	Transportation $O(n \log n)$	Yes
Total tardiness	$O(P \log P)$	Pseudopolynomial NP-hard	EDD $O(n \log n)$	No
Total weighted tardiness	Assignment $O(P^3)$	Strongly NP-hard	Quadratic transportation Polynomial	Yes

REFERENCES

COSMADAKIS, S. S., AND C. H. PAPADIMITRIOU. 1984. The Traveling Salesman Problem With Many Visits to Few Cities. *SIAM J. Comput.* **13**, 99–108.

DU, J., AND J. LEUNG. 1990. Minimizing Total Tardiness On One Machine is NP-Hard. *Math. Opns. Res.* **15**, 483–495.

FRENCH, S. 1982. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*. Ellis Horwood Ltd., Chichester, England.

HOCHBAUM, D. S., AND D. B. SHMOYS. 1987. Using Dual Approximation Algorithms for Scheduling Problems: Practical and Theoretical Results. *J. Assoc. Comput. Mach.* **34**, 144–162.

HOCHBAUM, D. S., AND R. SHAMIR. 1988. Strongly Polynomial Algorithms for the High Multiplicity Scheduling Problem. Technical Report 100/88. Institute of Computer Sciences, Tel Aviv University, Israel.

HOCHBAUM, D. S., AND R. SHAMIR. 1990. Minimizing the Number of Tardy Job Units Under Release Time Constraints. *Discrete Appl. Math.* **28**, 45–57.

HOCHBAUM, D. S., R. SHAMIR AND J. G. SHANTHIKUMAR. 1988. A Polynomial Algorithm for an Integer Quadratic Nonseparable Transportation Problem. Technical Report 113/88. Institute of Computer Sciences, Tel Aviv University, Israel. *Math. Prog.* (to appear).

HOFFMAN, A. J. 1963. On Simple Linear Programming Problems. In *Convexity: Proceedings of Symposia in Pure Mathematics*, Vol. 7, V. Klee (ed.). American Mathematical Society, Providence, R.I. 317–327.

IWANO, K., AND K. STEIGLITZ. 1987. Testing for Cycles in Infinite Graphs With Periodic Structure. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 46–55.

KARP, R. M. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, R. E. Miller, and J. W. Thatcher (eds.). Plenum Press, New York, 85–103.

LAWLER, E. L. 1976. Sequencing to Minimize the Weighted Number of Tardy Jobs. *Rev. Francaise Automat. Informat. Rec. Opnl. Ser. Bleue* **10.5** (Suppl.), 27–33.

LAWLER, E. L. 1977. A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness. *Anns. Discrete Math.* **1**, 331–342.

LAWLER, E. L., AND J. M. MOORE. 1969. A Functional Equation and Its Applications to Resource Allocation and Sequencing Problem. *Mgmt. Sci.* **16**, 77–84.

LENGAUER, T. 1982. The Complexity of Compacting Hierarchically Specified Layouts of Integrated Circuits. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, New York, 358–368.

LENAGAUER, T. 1986. Hierarchical Planarity Testing Algorithm. In *Proceedings of the 13th International Colloquium on Automata, Languages and Programming*. Rennes, France, L. Kott. (ed.). Spring-Verlag, New York, 215–225.

LENAGAUER, T. 1987. Efficient Algorithms for Finding the Minimum Spanning Forests of Hierarchically Defined Graphs. *J. Algorithms*, 260–284.

MOORE, J. M. 1968. An *n*-Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs. *Mgmt. Sci.* **15**, 102–109.

ORLIN, J. B. 1988. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. ACM Press, New York, 377–387.

PSARAFTIS, H. N. 1980. A Dynamic Programming Approach for Sequencing Groups of Identical Jobs. *Opns. Res.* **28**, 347–359.

ROTHKOPF, M. 1966. The Traveling Salesman Problem: On the Reduction of Certain Large Problems to Smaller Ones. *Opns. Res.* **14**, 532–533.