OPERATING SYSTEMS

deterministic scheduling
parallel machines
complexity
analysis of algorithm
linear programming

SYSTÈMES D'EXPLOITATION

ordonnancement déterministe
machines parallèles
complexité
programmation linéaire

Scheduling preemptible tasks on parallel processors with information loss

Ordonnancement de tâches préemptibles sur des processeurs parallèles avec perte d'information

Jacek BŁAŻEWICZ

Politechnika Poznánska ul. Piotrowo 3a, 60965 Poznan

Poznań, Poland

Jacek Błażewicz was born in Poznań, Poland, on August 11, 1951. He received the M. Sc., Ph. D. and Dr. habil. degrees (all in computer science) from the Technical University of Poznań, Poznań, in 1974, 1977 and 1980 respectively.
Since 1974 he has been with the Technical University of Poznań, where he is presently the Deputy Dean of the Electrical Engineering Faculty. His research interests are algorithms design and complexity analysis of algorithms especially in scheduling theory and in data transmission systems.
Dr. habil. Błażewicz is a member of the Polish Cybernetical Society, the Polish Computer Science Society and the American Mathematical Society.

COMMENTARY

J. Blazewicz discusses the problem of deterministic scheduling of critical date tasks on a set of processors. We know that this kind of problem is generally NP-complete. The form of analysis proposed considers the limiting case in which pre-emptive scheduling is used (a task being executed on a processor may be suspended at any moment in favour of another task), and in which the critical dates for tasks may be exceeded, with delays beyond deadline taken into account in an objective function which it is our aim to minimize.
In the first stage of his analysis, the author shows that the scheduling problem posed in this way may be reformulated as a minimal flow problem. He then goes on to transform this minimal flow problem into its equivalent linear programming problem, to show that it can be expressed by an algorithm with polynomial complexity. Finally, using Khachian's recently developed method, giving a solution algorithm for a linear programming problem which is polynomial in the number of variables and the number of constraints, he deduces that the initial scheduling problem is of polynomial complexity.
This article is a good illustration of a general approach used in algorithm complexity analysis : reformulate the initial problem, whose complexity is unknown, into an equivalent problem, whose complexity is known. In this way, work such as that published by Khachian in 1979, which shows that a linear program may be resolved by an algorithm with polynomial complexity, has major consequences for many problems, as long as they can be formulated as linear programs.

Dominique Potier

PRÉSENTATION

J. Blasewicz aborde dans cet article le problème de l'ordonnancement déterministe de tâches à dates critiques sur un ensemble de processeurs. On sait que ce type de problèmes est en général NP-complet. L'analyse proposée considère le cas plus restrictif où l'ordonnancement est préemptif (une tâche en cours d'exécution sur un processeur peut être suspendue à tout instant au profit d'une autre tâche), et où les dates critiques des

tâches peuvent être dépassées, le dépassement introduisant une pénalité prise en compte dans une fonction objectif que l'on cherche à minimiser.
Dans une première étape, l'auteur montre que le problème d'ordonnancement ainsi posé peut être reformulé comme un problème de flot minimal. Ensuite, afin de démontrer que ce problème de flot minimal peut être résolu par un algorithme de complexité polynomial, il est transformé en un programme linéaire équivalent. Enfin, en faisant appel à la méthode récente de Khachian, qui donne un algorithme polynomial sur le nombre de variables et le nombre de contraintes pour la résolution d'un programme linéaire, il en est déduit que le problème d'ordonnancement initial est de complexité polynomiale.
L'article illustre une démarche générale utilisée en analyse de complexité d'algorithmes : reformuler le problème initial, de complexité inconnue en un problème équivalent, de complexité connue. Ainsi, un résultat comme celui de Khachian, publié en 1979, qui démontre qu'un programme linéaire peut être résolu par un algorithme de complexité polynomiale, a-t-il des retombées beaucoup plus larges sur de nombreux problèmes, chaque fois que ces problèmes peuvent être formulés en programmes linéaires.

Dominique Potier

TABLE DES MATIÈRES

1. Introduction

The problems of scheduling tasks on processors (or machines) have been investigated for more than twenty years. We are thinking of deterministic scheduling problems in which ready times (or release dates) and execution times of all the tasks are known a priori : probabilistic problems were investigated much earlier. In the past few years, one can observe a great growth of interest in scheduling theory (we refer the reader to [Baker 74], [Coffman 76] and [Graham 77] for a survey of the results obtained). This growth follows the increasing application of computer systems and the need to work out methods of analysis which may yield suggestions to operating system designers. The deterministic approach is especially justified in process control system where all task parameters are usually known in advance, and if not, as for example execution times, upper bounds may be assumed.

To evaluate schedules several criteria have been used. These include schedule length, mean (or weighted) flow time and some criteria involving deadlines like maximum lateness, mean (or weighted) tardiness, or the number of late tasks. The first criterion is of value for computing centers, since its minimization leads to the maximization of processor utilization. The second criterion is important from the user's point of view since minimizing mean flow time also decreases the number of unfinished tasks in the system.

The criteria involving deadlines have been intensively studied, see e.g. [Błażewicz 76, 77], [Garey 76, 77], [Horn 74], [Jackson 55], [Labetoulle 74], [Liu 73]. They may be used in computer control systems, but rather only in cases when the value of a criterion in the optimal schedule is not greater than 0. If this value exceeds 0, the interpretation of the resulting schedule is economic rather than having an application in real computer control systems. This is because in the economic case penalties are introduced depending on the time at which the tasks are finished, while exceeding a deadline in the case of a control system, when for example collecting data from measurement points, causes the loss of all the uncollected data. Similarly, when taking sample measurements to obtain a mean value, the exceeding of a deadline causes the complete loss of samples not yet made, and thus reduces the precision of the estimate. Thus, in computer control systems in which the deterministic approach is most strongly justified, we would rather like to minimize the mean (or mean weighted) information loss caused by the loss of those parts of tasks unprocessed at their deadlines, no matter when they are completed. This is a new criterion which we will be concerned with in this paper.

When analyzing scheduling algorithms which are to be applied directly in operating systems, attention should be paid to their computational complexity. It is rather obvious that only algorithms whose running time is bounded from above by a polynomial in the input length, may be used. Thus, given a problem of scheduling tasks, one first attempts to construct an optimization, polynomial-in-time algorithm for solving the problem. Using, however, the theory of NP-completeness (see e.g. [Garey 79]), one can prove some problems to be NP-complete, thus, unlikely to admit such algorithms. This type of analysis is used below where we analyze the complexity of the problem of scheduling tasks on parallel processors to minimize mean information loss.

The model of the computing system which we are concerned with and the basic definitions will be described in Section 2. In Section 3 the problem of minimizing the mean information loss for the case of nonpreemptible tasks will be shown to be NP-complete in the strong sense, even for the one processor. Then we will present a polynomial-in-time method for solving the problem of

the preemptive scheduling of tasks on parallel identical processors to minimize mean weighted information loss by means of a linear programming approach. We extend this method in Section 4 to solve the problem of scheduling tasks on a fixed number of uniform processors.

2. Some definitions

We will be considering a computing system, two components of which are : a set of *m parallel processors* $\{ P_1, P_2, ..., P_m \}$ and a set of *n tasks* $\{ T_1, T_2, ..., T_n \}$. The processors are assumed to be either *identical* with equal processing speeds or to be *uniform* where processor P_i has its processing speed equal to s_i. Each processor is capable of processing at most one task at a time. Each task T_j is characterized by the following parameters : *processing time* p_j (in the case of uniform processors this is a standard processing time and to process task T_j on processor P_i takes p_j/s_i time units), *ready time* r_j, *deadline* d_j, $d_j > r_j$, and *weight* w_j. Tasks are assumed to be *independent*, that is no precedence constraints among them exist.

Some other definitions will be useful. By *preemptible* we mean tasks that may be preempted at any moment and restarted later (perhaps on another processor) with no time losses. Tasks that cannot be preempted are called *nonpreemptible*. A *feasible schedule* is a specification of the assignment of processors to tasks and this assignment must satisfy the following conditions :

— all tasks are processed to completion :
— at most m tasks are processed at a time ;
— each task T_j is processed in the time interval $[r_j, \infty]$. For every task T_j in the schedule we denote by Y_j its *information loss*, defined as follows :

$$Y_j = \begin{cases} \text{the amount of processing of } T_j \text{ exceeding } d_j, \\ \text{if } T_j \text{ is not completed on time,} \\ O, \text{ otherwise.} \end{cases}$$

To evaluate schedules we will use the following performance measures : *mean information loss* $\overline{Y} = \sum_{j=1}^{n} Y_j/n$, and *mean weighted information loss* $\overline{Y}_w = \sum_{j=1}^{n} w_j Y_j/n$. The schedule will be called *optimal* if the value of the respective performance measure is minimal for it.

A *scheduling algorithm* is a step by step procedure that produces a schedule for every given set of tasks. By an *optimization scheduling algorithm* we mean an algorithm which always finds an optimal schedule. We assume that the reader is familiar with the basic concepts of the computational complexity of decision problem, described for example in [Garey 79].

3. Scheduling on Identical Machines

When considering nonpreemptive scheduling of tasks to minimize mean information loss one may notice that the problem (its decision version) is strongly NP-complete. This may be proved by using a proof similar to the one presented in [Garey 79], where the strong NP-completeness of the problem of minimizing mean tardiness (decision version) has been proved. It follows that no polynomial-in-time (and also pseudopolynomial-in-time) algorithm is likely to exist for these problems.

However, this is not the case when considering preemptible tasks. We now present an algorithm which allows us to solve in polynomial time the problem of scheduling these tasks on identical processors in order to minimize mean weighted information loss. We will formulate the problem as a special network flow problem, then we will solve it by means of a linear programming method.

Let $G(V, E)$, where $V = \{ s_1, s_2, v_1, v_2, ..., v_p \}$ is a set of vertices and $E = \{ e_1, e_2, ..., e_q \}$ is a set of arcs, be a network. With each arc e_i, $i = 1, 2, ..., q$, of this network there is associated an interval $[f_i, g_i]$, and a valid flow through this arc, ϕ_i, must satisfy $f_i \leqslant \phi_i \leqslant g_i$, where g_i is called the capacity of arc e_i. To reformulate our scheduling problem as a problem of finding a flow pattern with certain desired features in the above network, let us assume that there exist $l \leqslant 2n$ different deadlines and ready times, which we can denote by $a_1, a_2, ..., a_l$. Let us assume moreover, that they are ordered such that $a_1 < a_2 < \cdots < a_l$. Before giving the formulation more formally we will describe briefly the idea behind it. We may distinguish four groups of vertices in the network (*fig.* 1). The first represents time intervals in the schedule. This means that there exist vertices corresponding to $[a_0, a_1]$, $[a_1, a_2]$, ..., $[a_{l-1}, a_l]$, $[a_l, a_{l+1}]$ where $a_0 = 0$ and the last interval corresponds to the " rest " of processing, and a_{l+1} is large enough to guarantee the completion of all the tasks, e.g. $a_{l+1} = \max_j \{ r_j \} + \sum_{j=1}^{n} p_j$. The capacities of arcs joining the source of a network s_1 with these vertices are equal to the processing capabilities of all the processors in the corresponding intervals, i.e. they are equal to m times the lengths of the intervals.

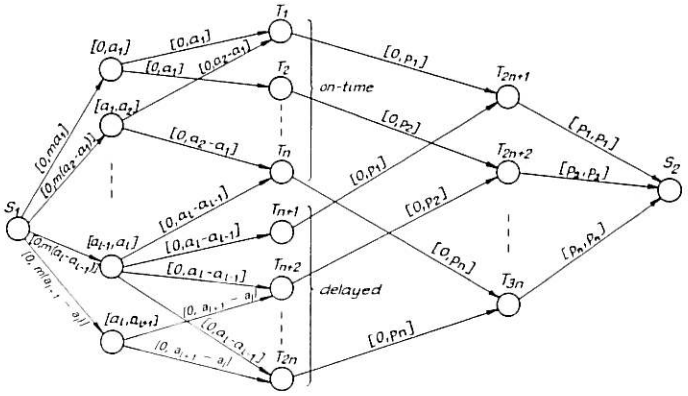

Figure 1. — Four groups of vertices in the network.

The second group of vertices $\{ T_1, T_2, ..., T_n \}$ represents the tasks (or parts of them) processed on-time. Thus, vertex $[a_i, a_{i+1}]$ $i = 0, 1, 2, ..., k$, is joined by an arc to vertex T_j $j = 1, 2, ..., n$ if and only if $r_j \leqslant a_i$ and $d_j \geqslant a_{i+1}$. The capacity of such an arc is equal to the processing capability of a single processor in this interval, that is to $a_{i+1} - a_i$.

The third group of vertices $\{ T_{n+1}, T_{n+2}, ..., T_{2n} \}$ represents tasks (or parts of them) which exceed their deadlines. The vertex $[a_i, a_{i+1}]$ is joined to the vertex T_{n+j} if and only if $d_j \leqslant a_i$. The capacity of such an arc is the same as in the previous case.

The fourth group of vertices $\{ T_{2n+1}, T_{2n+2}, ..., T_{3n} \}$ is designed to ensure the completion of all the tasks. Vertices T_j and T_{n+j}, $j = 1, 2, ..., n$, are joined by arcs of capacities equal to p_j, to vertex T_{2n+j}. The flow outgoing from the latter must be equal exactly to p_j.

The objective is to find in such a network a feasible flow pattern for which the weighted sum of flows outgoing from the third group of vertices is minimum, thus, minimizing mean weighted information loss.

We are now prepared to give a more formal description of the network formulation of the scheduling problem. Given the latter as described at the beginning of this Section, the corresponding network $G(V, E)$ may be defined as follows. Let $V = \{ s_1, s_2 \} \cup \{ [a_i, a_{i+1}] : i = 0, 1, ..., l \} \cup \{ T_j : j = 1, 2, ..., n \} \cup \{ T_{n+j} : j = 1, 2, ..., n \} \cup \{ T_{2n+j} : j = 1, 2, ..., n \}$. Then let $E = \{ (s_1, [a_i, a_{i+1}]), i = 0, 1, ..., l$; with a capacity of the arc equal to $m(a_{i+1} - a_i) \} \cup \{ ([a_i, a_{i+1}], T_j)$, for each i and j for which $r_j \leqslant a_i$ and $d_j \geqslant a_{i+1}$; with a capacity equal to $a_{i+1} - a_i \} \cup \{ ([a_i, a_{i+1}], T_{n+j})$, for each i and j for which $d_j \leqslant a_i$; with a capacity equal to $a_{i+1} - a_i \} \cup \{ (T_j, T_{2n+j}), j = 1, 2, ..., n$: with a capacity equal to $p_j \} \cup \{ (T_{n+j}, T_{2n+j}), j = 1, 2, ..., n$: with a capacity equal to $p_j \} \cup \{ (T_{2n+j}, s_2), j = 1, 2, ..., n$: with a capacity equal to p_j and a lower bound equal also to $p_j \}$. The lower bounds of all but the last group of arcs are equal to 0. The objective is to find a feasible flow pattern for which $\sum_{j=1}^{n} w_j \phi_{(T_{n+j}, T_{2n+j})}$ is minimal.

It is quite clear that by solving the above network flow problem we find also an optimal schedule since $\sum_{j=1}^{n} w_j \phi_{(T_{n+j}, T_{2n+j})}$ is equal to the weighted sum of the processing times of those parts of tasks which are unprocessed at their deadlines. Thus, the assignment of tasks to processors in a optimal schedule is given by the obtained values of $\phi_{([a_i, a_{i+1}], T_k)}$, $i = 0, 1, ..., l$, $k = 1, 2, ..., 2n$.

To show that the above network flow problem may be solved in polynomial time we reformulate it as a linear programming problem. The latter can be stated as follows.

Minimize

$$\sum_{j=1}^{n} w_j \sum_{i \in W_j} \phi_{([a_i, a_{i+1}], T_{n+j})} \qquad (1)$$

where W_j is the set of all i for which $d_j \leqslant a_i$. Subject to

$$\sum_{\{j : i \in Z_j\}} \phi_{([a_i, a_{i+1}], T_j)} + \sum_{\{j : i \in W_j\}} \phi_{([a_i, a_{i+1}], T_{n+j})} \leqslant \leqslant m(a_{i+1} - a_i) \quad i = 0, 1, ..., l. \qquad (2)$$

where Z_j is the set of all i for which $r_j \leqslant a_i$ and $d_j \geqslant a_{i+1}$

$$\phi_{([a_i, a_{i+1}], T_j)} \leqslant a_{i+1} - a_i, \quad j = 1, 2, ..., n, \quad i \in Z_j \qquad (3)$$

$$\phi_{([a_i, a_{i+1}], T_{n+j})} \leqslant a_{i+1} - a_i, \quad j = 1, 2, ..., n, \quad i \in W_j \qquad (4)$$

$$\sum_{i \in Z_j} \phi_{([a_i, a_{i+1}], T_j)} + \sum_{i \in W_j} \phi_{([a_i, a_{i+1}], T_{n+j})} = p_j, \quad j = 1, 2, ..., n. \qquad (5)$$

The number of variables in the above linear programming problem is $0(n^2)$ and the number of constraints is also $0(n^2)$. To solve the above problem one can use the recently developed method of [Khachian 79] (for detailed description of Khachian's algorithm see for example [Gacs 79], [Korte 81]). This method is polynomial in the number of variables and constraints. Thus, it enables us to solve our problem of scheduling preemptible tasks in order to minimize mean weighted information loss in time bounded from above by a polynomial in the number of tasks and processors.

4. Extension to the Uniform Processor Case

In this section we extend the model of section 3 to cover the case of uniform processors. Let us recall that in this case task processing requirements are represented by their standard processing times p_j, $j = 1, 2, ..., n$. Moreover, processors differ from each other by their processing speeds s_i, $i = 1, 2, ..., m$. Thus, the time needed to process task T_j, $j = 1, 2, ..., n$, on processor P_i, $i = 1, 2, ..., m$, is equal to p_j/s_i. Let us assume that the processors are ordered in such a way that $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_m$. It is obvious that in each time interval $[a_i, a_{i+1}]$, $i = 0, 1, ..., l$, the amount of processing done by the processors cannot exceed $\sum_{k=1}^{m} s_k(a_{i+1} - a_i)$. Thus, the inequalities (2) now have the form :

$$\sum_{\{j : i \in Z_j\}} \phi_{([a_i, a_{i+1}], T_j)} + \sum_{\{j : i \in W_j\}} \phi_{([a_i, a_{i+1}], T_{n+j})} \leqslant \leqslant (a_{i+1} - a_i) \sum_{k=1}^{m} s_k \quad i = 0, 1, ..., l. \qquad (2')$$

On the other hand, a task processed in the time interval $[a_i, a_{i+1}]$ cannot obtain more processing than $s_1(a_{i+1} - a_i)$, two tasks more than $(s_1 + s_2)(a_{i+1} - a_i)$ and so on, and m or more tasks, more than $\sum_{k=1}^{m} s_k(a_{i+1} - a_i)$. Thus, we have to change the constraints (3) and (4), to ensure the fulfilment of the above conditions for every possible combination of tasks processed in parallel.

$$\phi_{([a_i, a_{i+1}], T_k)} \leqslant s_1(a_{i+1} - a_i), \quad k = 1, 2, ..., 2n,$$

if $k \leqslant n$ then $i \in Z_k$, otherwise $k = n + j$ and $i \in W_j$:

$$\sum_{k \in K_2} \phi_{([a_i, a_{i+1}], T_k)} \leqslant (s_1 + s_2)(a_{i+1} - a_i) \quad i = 0, 1, ..., l,$$

$K_2 \subset \{ 1, 2, ..., 2n \} \wedge | K_2 | = 2$: if $k \leqslant n$ then $i \in Z_k$, otherwise $k = n + j$ and $i \in W_j$ (3')

$$\sum_{k \in K_m} \phi_{([a_i, a_{i+1}], T_k)} \leqslant (a_{i+1} - a_i) \sum_{r=1}^{m} s_r, \quad i = 0, 1, ..., l,$$

$K_m \subset \{ 1, 2, ..., 2n \} \wedge | K_m | = m$: if $k \leqslant n$ then $i \in Z_k$, otherwise $k = n + j$ and $i \in W_j$.

The equations (5) as well as the criterion (1) remain the same. Thus, to find an optimal schedule (i.e. one minimizing the mean weighted information loss) one should solve the linear programming problem (1), (2'), (3') and (5). Again, the optimal parts of tasks which are to be processed in each time interval, are given by the values $\phi_{([a_i,a_{i+1}],T_j)}$ and $\phi_{([a_i,a_{i+1}],T_{n+j})}$, $j = 1, 2, \ldots, n$, $i = 0, 1, \ldots, l$. But now, to obtain a schedule, one should (in each time interval) apply an algorithm for scheduling preemptible tasks on uniform processors in order to minimize schedule length [Horvath 77].

Let us now check the computational complexity of the above approach. The number of variables remains unchanged as compared with the formulation in section 3. However, the number of constraints is greater. Namely, it may be computed that it is $0(n^m)$, thus for fixed m it is bounded from above by a polynomial in the number of tasks. It follows that using Khachian's method we can solve the problem of scheduling preemptible tasks on a fixed number of uniform processors to minimize mean weighted information loss in time bounded from above by a polynomial in the number of tasks.

To evaluate the above approach, numerical experiments were also conducted. Since the mean behavior of the simplex algorithm is better than that of the Khachian's one ([1]) (see for example [Papadimitriou 82]) the first algorithm was chosen for this experiment. The specific version used was based upon the product form of the inverse. The experiment was carried on an Odra-1305 computer (compatible with ICL 1900). Task parameters were drawn from a uniform distribution. Computation times for networks, representing scheduling problem for uniform processors and consisting of about 20 nodes varied from about 1 minute to 4 minutes : for networks of about 50 nodes, from 8 minutes to 30 minutes and for networks of about 100 nodes, from 20 minutes to 90 minutes. Variance in computation times for a given size of the network appeared to be primarily a function of the number of processors (i.e. the number of arcs in the network) and the values of processing times.

5. Conclusions

We have considered the problem of scheduling tasks in order to minimize mean weighted information loss. As was mentioned this criterion may be of value in process control systems. The problem is NP-complete for the nonpreemptive case, even for one processor and equal weights. However, it is solvable in polynomial time when preemptions are allowed in the case of arbitrary number of identical processors and also when fixed number of uniform processors is considered.

It would be worthwile to extend this result to the case of unrelated processors, however, no simple extension seems to be possible. It might also be of value to consider an extended model of a computing system by introducing additional resources, other than processors, as for example in [Błażewicz 79], [Garey 75], [Słowiński 81], [de Werra 82].

([1]) Despite the fact that the worst case behavior of the simplex algorithm is worse than that of the Khachian's procedure and moreover, it cannot be bounded from above by a polynomial in the input length.

To this end let us also comment briefly on the relations of the mean weighted information loss with other criteria when preemptible tasks are considered. A network flow approach is used in [Horn 74] to test whether or not there exists a schedule with no task late. Applying then binary search procedure, one may also find a schedule that minimizes maximum lateness for a set of identical processors. Moreover, another linear programming approach may be used to find such a schedule even for an unrelated processor case [Lawler 78] and [Słowiński 81]. On the other hand, when scheduling to minimize mean tardiness is considered, the complexity of the problem remains an open question for all the processor types. The approach presented in this paper cannot be used in this case. Minimizing mean weighted tardiness is a much harder problem. It is NP-complete in the strong sense even for the one processor case. Following these remarks one may say that from the view-point of the complexity of scheduling problems (their hardness) mean weighted information loss lies in between maximum lateness and mean tardiness.

REFERENCES

[Aho 74] A. V. Aho, J. E. Hopcroft, J. D. Ullman : *The Design and Analysis of Computer Algorithms ;* Addison-Wesley, Reading, Mass, 1974.

[Baker 74] K. R. Baker : *Introduction to Sequencing and Scheduling ;* J. Wiley and Sons, New York, 1974.

[Błażewicz 76] J. Błażewicz : *Scheduling Dependent Tasks with Different Arrival Times to Meet Deadlines,* in E. Gelenbe, H. Beilner (eds.) Modelling and Performance Evaluation of Computer Systems ; North Holland, Amsterdam, 1976, 57-65.

[Błażewicz 77] J. Błażewicz : *Simple Algorithms for Multiprocessor Scheduling to Meet Deadlines ;* Information Processing Letters **6** (3), 1977, 162-164.

[Błażewicz 79] J. Błażewicz : *Deadline Scheduling of Tasks with Ready Times and Resource Constraints ;* Information Processing Letters **8,** (2), 1979, 60-63.

[Coffman 76] E. G. Jr Coffman (ed.) : *Computer and Job-Shop Scheduling Theory ;* J. Wiley, New York, 1976.

[Gác 81] P. Gács, L. Lovász : *Khachian's Algorithm for Linear Programming* in M. König, B. Korte, K. Ritter (eds.) ; Mathematical Programming Studies **14,** 1981, 61-68.

[Garey 75] M. R. Garey, D. S. Johnson : *Complexity Results for Multiprocessor Scheduling under Resource Constraints ;* SIAM. J. on Computing **4,** 1975, 397-411.

[Garey 76] M. R. Garey, D. S. Johnson : *Scheduling Tasks with Nonuniform Deadlines on Two Processors ;* Journ. ACM **23** (3), 1976, 461-467.

[Garey 77] M. R. Garey, D. S. Johnson : *Two Processors Scheduling with Start-Times and Deadlines ;* SIAM Journ. on Computing **6,** 1977, 416-426.

[Garey 78] M. R. Garey, D. S. Johnson : "*Strong*" *NP-completeness Results : Motivation, Examples and Implications ;* Journ. ACM **25,** 1978, 499-508.

[Garey 79] M. R. Garey, D. S. Johnson : *Computers and Intractability : A Guide to the Theory of NP-Completeness ;* W. H. Freeman, San Francisco, 1979.

[Graham 77] R. L. Graham, E. L. Lawler *et al. : Optimization and Approximation in Deterministic Sequencing and Scheduling Theory : a Survey ;* Report BW **82**/77, Stichting Mathematisch Centrum, 1977.

[Horn 74] W. A. HORN : *Some Simple Scheduling Algorithms ;* Naval Res. Logist. Quart. **21,** 1974, 177-185.

[Horvath 77] E. G. HORVATH, S. LAM, R. SETHI : *A Level Algorithm for Preemptive Scheduling ;* Journ. ACM **24** (1), 1977.

[Jackson 55] J. R. JACKSON : *Scheduling a Production Line to Minimize Maximum Tardiness ;* Res. Rep. 43, Management Research Project, University of California, Los Angeles, 1955.

[Khachian 79] L. G. KHACHIAN : *A Polynomial Algorithm for Linear Programming ;* Doklady Akad. Nauk USSR 244, 1979.

[Korte 81] B. KORTE (ed.) : *Modern Applied Mathematics, Optimization and Operations Research ;* North Holland, 1981.

[Labetoulle 74] J. LABETOULLE : *Some Theorems on Real Time Scheduling* in E. Gelenbe, R. Mahl (eds.) *Computer Architecture and Networks ;* North Holland, Amsterdam, 1974, 285-298.

[Lawler 78] E. L. LAWLER, J. LABETOULLE : *On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming ;* Journ. ACM **25,** 1978, 612-619.

[Liu 73] C. L. LIU, J. W. LAYLAND : *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment ;* Journ. ACM **20** (1), 1973, 46-61.

[Papadimitriou 82] Ch. PAPADIMITRIOU, K. STEIGLITZ : *Combinatorial Optimization ;* Prentice Hall, N.J. 1982.

[Słowiński 81] R. SŁOWIŃSKI : *L'ordonnancement des tâches préemptives sur les processeurs indépendants en présence de ressources supplémentaires ;* RAIRO Informatique **15,** 1981, 155-166.

[Werra 82] D. DE WERRA : *Network Flows, Linear Programming and Scheduling ;* Report : Ecole Polytechnique Fédérale de Lausanne, 1982.

Article reçu le 16 avril 1982. Version révisée le 23 mai 1984.