Lower Bounds for On-line Scheduling with Precedence Constraints on Identical Machines

Leah Epstein

Tel-Aviv University *

Abstract. We consider the on-line scheduling problem of jobs with precedence constraints on m parallel identical machines. Each job has a time processing requirement, and may depend on other jobs (has to be processed after them). A job arrives only after its predecessors have been completed. The cost of an algorithm is the time that the last job is completed. We show lower bounds on the competitive ratio of on-line algorithms for this problem in several versions. We prove a lower bound of $2-1/m$ on the competitive ratio of any deterministic algorithm (with or without preemption) and a lower bound of $2-2/(m+1)$ on the competitive ratio of any randomized algorithm (with or without preemption). The lower bounds for the cases that preemption is allowed require arbitrarily long sequences. If we use only sequences of length $O(m^2)$, we can show a lower bound of $2-2/(m+1)$ on the competitive ratio of deterministic algorithms with preemption, and a lower bound of $2-O(1/m)$ on the competitive ratio of any randomized algorithm with preemption. All the lower bounds hold even for sequences of unit jobs only. The best algorithm that is known for this problem is the well known List Scheduling algorithm of Graham. The algorithm is deterministic and does not use preemption. The competitive ratio of this algorithm is $2-1/m$. Our randomized lower bounds are very close to this bound (a difference of $O(1/m)$) and our deterministic lower bounds match this bound.

1 Introduction

We consider the problem of scheduling a sequence of n jobs on m parallel identical machines. There are precedence constraints between the jobs, which can be given by a directed acyclic graph on the jobs. In this graph each directed edge between jobs j_1 and j_2 indicates that j_1 has to be scheduled before j_2. We consider an on-line environment in which a job is known only after all its predecessors in the graph are processed by the on-line algorithm. Each job j has a certain time requirement w_j (which is known when the job arrives). The cost of an algorithm is the makespan, which is the time in which the last job is completed. This model is realistic since often the running times of specific jobs are known in advance,

* Dept. of Computer Science, Tel-Aviv University. E-Mail: lea@math.tau.ac.il.

but it is unknown whether after these jobs, there will be need to perform jobs that depend on some previous jobs.

We compare the performance of on-line algorithms and the optimal off-line algorithm that knows the sequence of jobs and the dependencies graph in advance. We use the competitive ratio to measure the performance of the algorithm.

Denote the cost of the on-line algorithm by C_{on} and the cost of the optimal off-line algorithm by C_{opt}. The competitive ratio of an on-line algorithm is r if for any input sequence of jobs: $C_{on} \leq r \cdot C_{opt}$.

We consider a several models. Consider a job j which is released at time t_1 (t_1 is the time that its last predecessor finished running), and has processing time requirement w_j. In the model without preemption, j has to be processed on one machine for w_j time units, starting at some time t, $t \geq t_1$ till time $t + w_j$. In the model that allows preemption, each running job can be preempted, i.e. its processing may be stopped and resumed later on the same, or on different machine. Thus j still has to be processed for a total of w_j time units, but not necessarily continuously, or on one machine (it cannot be processed on more than one machine at the same time). The algorithms may be either deterministic or randomized. For randomized algorithms the competitive ratio is r if for any input sequence of jobs: $E(C_{on}) \leq r \cdot C_{opt}$. It is also possible to consider special sequences as sequences that consist only of unit jobs, and sequences of bounded length.

Related problems and results: The problem of scheduling a set of tasks on parallel machines has been widely studied in many variants. In the basic problem, a sequence of jobs is to be scheduled on a set of identical parallel machines. The jobs may be independent or have precedence constraints between them. They may all arrive at the beginning or have release times, or arrive according to the precedence constraints. The running times of the jobs can be known in advance (at arrival time) or unknown (till they are finished). It is also possible to allow restarts (a job is stopped, and resumed later from the beginning) or to allow preemptions. The goal is to construct a schedule of minimum length. (There are variants with other cost functions too). All those problems, in their off-line version are NP-hard [6].

The first one to introduce the on-line scheduling problem was Graham [8, 9]. He also introduced the algorithm List Scheduling. This algorithm, each time that some machine is idle, schedules a job that is available (if there exists such a job which was not scheduled yet and already arrived). List Scheduling suits all the above mentioned cases, and has the competitive ratio of $2 - 1/m$. In this paper we show that for our problem, the algorithm is optimal in the deterministic case, and that it is almost optimal for randomized algorithms.

Our deterministic lower bounds build on the paper of Shmoys, Wein and Williamson [15]. They consider the problem of scheduling a sequence of independent tasks on-line. The duration of a job in unknown until it is completed, but there are no precedence constraints between the jobs. They show a lower bound of $2 - 1/m$ on the competitive ratio of any deterministic algorithm without preemption. In this paper we adapt this lower bound to a lower bound of

$2 - 1/m$ for our problem. We also build our lower bounds for deterministic algorithms with preemption on some of the ideas of their lower bound. They also show that the same lower bound holds with preemption, and they show a lower bound of $2 - O(1/\sqrt{m})$ on the competitive ratio of randomized algorithms without preemption. The last lower bound can be also adapted to our problem (even with preemption), but we show a much stronger lower bound for randomized algorithms.

Our results: All the results apply even if the sequences may consist of unit jobs only. Also the number of jobs, the structure, and makespan of the optimal off-line assignment is known in advance in all lower bounds.

We prove the following lower bounds for deterministic algorithms:

- A lower bound of $2 - 1/m$ on the competitive ratio of any deterministic on-line algorithm without preemption (even if the length of the sequence is limited to $O(m^2)$).
- A lower bound of $2 - 2/(m+1)$ on the competitive ratio of any deterministic on-line algorithm that allows preemption (even if the length of the sequence is limited to $O(m^2)$).
- A lower bound of $2 - 1/m$ on the competitive ratio of any deterministic on-line algorithm that allows preemption.

We prove the following lower bounds for randomized algorithms

- A lower bound of $2 - 2/(m+1)$ on the competitive ratio of any randomized on-line algorithm without preemption (even if the length of the sequence is limited to $O(m^2)$).
- A lower bound of $2 - O(1/m)$ on the competitive ratio of any randomized on-line algorithm that allows preemption (even if the length of the sequence is limited to $O(m^2)$).
- A lower bound of $2 - 2/(m+1)$ on the competitive ratio of any randomized on-line algorithm that allows preemption.

The similar results for all the models show that for this problem, neither randomization nor preemption can help in reducing the competitive ratio. Any algorithm would have the competitive ratio of $2 - O(1/m)$ which is very close to the competitive ratio of List Scheduling.

More related work: A summary on results for many variants of on-line scheduling problems appears in [14]. Results for the case that jobs arrive over time appear in [1, 5, 15, 17]. Note that [1, 17] show an algorithm with competitive ratio 3/2 even without preemption, for the case that jobs are independent, and the durations are known in advance (unlike our case in which there is a lower bound of $2 - O(1/m)$ even with preemption). Moreover, if preemption is allowed, there exists a 1-competitive algorithm [7, 10, 13, 16]. Results on scheduling with precedence constraints, but for other types of machine sets or jobs can be found in [3, 4, 11, 12]. There are also some off-line results with precedence constraints in [2]. Note that already for a related set of machines (different speeds) the problem of scheduling with precedence constraints is hard (competitive ratio of $\Omega(\sqrt{m})$ [3, 11]).

Structure of the paper: In section 2 we show lower bounds for deterministic algorithms. In section 3 we show lower bounds for randomized algorithms.

2 Deterministic Algorithms

In this section we show a lower bound of $2 - 1/m$ on the competitive ratio of deterministic algorithms with preemption. We give all the lower bounds in this section in detail, since some of the ideas in the randomized lower bounds build on the deterministic case. We begin with a simple lower bound without preemption. All lower bounds use sequences of unit jobs only.

Theorem 1. *The competitive ratio of any deterministic algorithm without preemption is at least* $2 - \frac{1}{m}$. *This is true even if all jobs are unit jobs.*

Proof. We use the following sequence: (all jobs are unit jobs). First $m(m-1)+1$ jobs arrive. Consider the on-line assignment. Since the total time to schedule $m(m-1)$ jobs is at least $m-1$ units of time, there is at least one job j assigned at time $m-1$ or later, and finishes at time m or later.

After that, a sequence of $m-1$ jobs $j_1, ..., j_{m-1}$ arrives. In this sequence the first job j_1 depends on j, and each job j_i depends on the previous one j_{i-1}; $(2 \leq i \leq m-1)$. It takes more $m-1$ units of time to schedule them and thus $C_{on} = m + (m-1) = 2m - 1$.

The optimal off-line algorithm would schedule j at time 0, and each j_i at time i and thus can finish all m^2 jobs in m time units. Thus $C_{opt} = m$ the competitive ratio is $2 - 1/m$.

We use a longer sequence to show that the same lower bound holds with preemption too.

Theorem 2. *The competitive ratio of any deterministic algorithm with preemption is at least* $2 - \frac{1}{m}$. *This is true even if all jobs are unit jobs.*

Proof. For an integer k, $(m^k+1)(m-1)+1$ jobs arrive. The minimum time to run those jobs even with preemption is at least $(m^{k+1}+m-m^k)/m = m^k+1-m^{k-1}$, thus for the on-line algorithm there is at least one job j that finishes at time at least $m^k + 1 - m^{k-1}$.

After those jobs, a sequence J of m^k jobs, that the first one depends on j, and each job depends on the previous one arrives. The time to run those jobs is at least m^k and thus $C_{on} = m^k + 1 - m^{k-1} + m^k = 2m^k - m^{k-1} + 1$.

The optimal off-line algorithm would schedule j at time 0, and since there are $m(m^k+1)$ jobs, the total time to run them would be $C_{opt} = m^k + 1$ (it is possible to run j and all jobs of J in $m^k + 1$ time units). The competitive ratio is $2 - \frac{m^{k-1}+1}{m^k+1} = 2 - \frac{1}{m} - \varepsilon_k$ where $\varepsilon_k \to 0$ when $k \to \infty$. Thus the competitive ratio of any deterministic on-line algorithm that allows preemption is at least $2 - 1/m$.

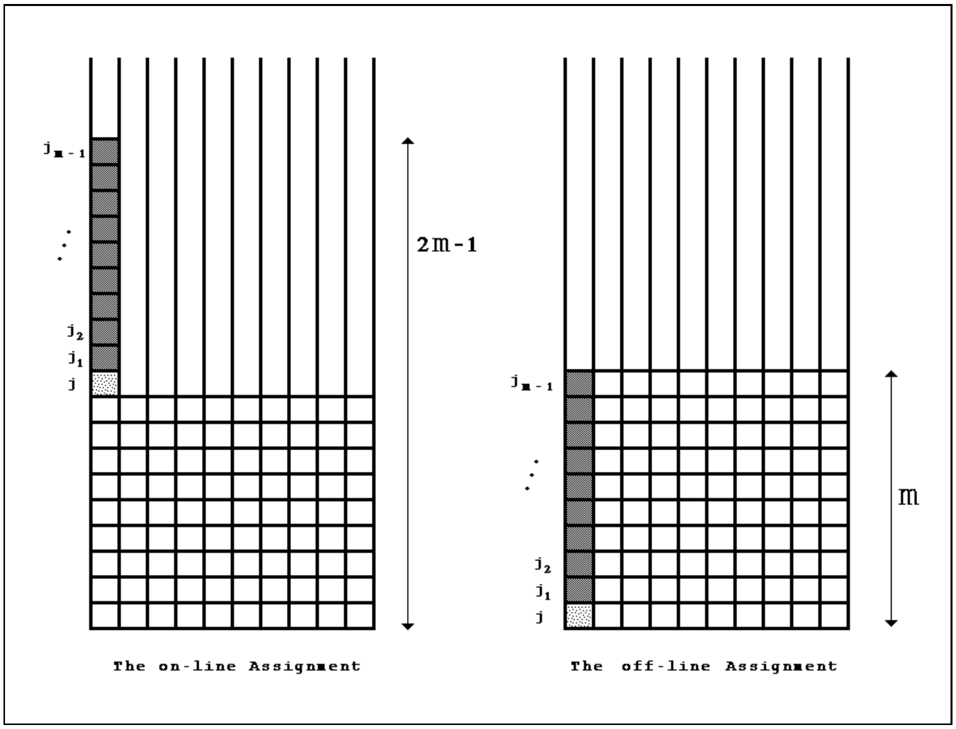

Fig. 1. The on-line and off-line assignments for the sequence in Theorem 1

now we show that the strength of the lower bound almost does not depend on the length of the sequence.

Theorem 3. *The competitive ratio of any deterministic algorithm with preemption is at least* $2 - \frac{2}{m+1}$. *This is true even if all jobs are unit jobs. And the length of the sequence is* $O(m^2)$.

Proof. We use the sequence from Theorem 2 with $k = 1$.

3 Randomized Algorithms

In all the proofs in this section, which are lower bounds on the competitive ratio of randomized algorithms we use an adaptation of Yao's theorem for on-line algorithms. It states that a lower bound for the competitive ratio of deterministic algorithms on any distribution on the input is also a lower bound for randomized algorithms and is given by $E(C_{on}/C_{opt})$. We will use only sequences for which C_{opt} is constant and thus in our case $E(C_{on}/C_{opt}) = E(C_{on})/C_{opt}$.

We begin with a lower bound without preemption. note that in this section the lower bound without preemption uses a totally different sequence than the lower bound with preemption. It is possible the same sequence as in the first proof in this section (Theorem 4) to get the lower bound of $2 - 1/m$ for deterministic algorithms without preemption. Note that here also all lower bound sequences consist of unit jobs only.

Theorem 4. *The competitive ratio of any randomized algorithm without preemption is at least $2 - \frac{2}{m+1}$. This is true even if all jobs are unit jobs.*

Proof. First $m-1$ phases of $m+1$ jobs arrive. In each phase, all jobs depend on m jobs of the previous phase (For each phase, the subset of m jobs from the previous phase is chosen among all $\binom{m+1}{m} = m+1$ possible subsets of m jobs with equal probability). After all $m^2 - 1$ jobs have arrived, another job that depends on m jobs of the last phase arrives (here also the m jobs are chosen among the $m+1$ possible subsets with equal probability).

For each phase i, $0 \leq i \leq m-2$, the optimal off-line algorithm schedules the m jobs that the next phase depends on them at time i. all other jobs are scheduled at time $m-1$ and thus $C_{opt} = m$.

Note that a phase can arrive only after all m jobs from the previous phase, that this phase depends on them were scheduled (the important jobs in this phase). The time to schedule a phase is 1 or 2 units of time, since if the correct subset of m jobs is chosen, it is possible to assign the last job later and use only one time unit. If the wrong subset was chosen, the algorithm has to use another unit of time to complete the running of the important jobs of this phase and uses two units of time. The last job requires exactly one unit of time. Each of the $m-1$ phases is placed correctly with probability $1/(m+1)$. For each phase, the probability to use a second unit of time is at least $m/(m+1)$. Thus the expectation of the on-line cost is at least $E(C_{on}) \geq m + (m-1)(m/(m+1))$, $C_{opt} = m$ the competitive ratio is at least $2m/(m+1) = 2 - 2/(m+1)$.

Now we show the simple lower bound for randomized algorithms with preemption. This lower bound uses a short sequence.

Theorem 5. *The competitive ratio of any randomized algorithm with preemption is at least $2 - O(\frac{1}{m})$. This is true even if all jobs are unit jobs, and the length of the sequence is $O(m^2)$.*

Proof. First m^2 jobs arrive. More m jobs $j_1, ..., j_m$ arrive so that j_1 depends on a subset J of m of the m^2 jobs, which is chosen uniformly at random among all $\binom{m^2}{m}$ possible subsets, and for $1 \leq i \leq m-1$, j_{i+1} depends on j_i.

Even with preemption, since each job requires one processing unit, and jobs cannot run simultaneously, no jobs can finish before time 1 and at most m jobs can finish at time 1. In general, at most im jobs can finish before time $i+1$. Let $b_1, ..., b_{m^2}$ be the set of the first m^2 jobs sorted according to their finishing time (b_1 finishes first). For $1 \leq i \leq m$, let J_i be the set $b_{im-m+1}, ..., b_{im}$. Each job in a set J_i cannot finish before time i. Let I be the set of indices i such that there is at least one job of J in J_i. Let i_1 be the maximum index in I. We define p_i to be the probability that $i = i_1$ for each $1 \le i \le m$. For $0 \le i \le m$, let q_i be the probability that all m jobs of J are chosen among the jobs in the sets $J_1, ..., J_i$ then $q_i = \binom{mi}{m} / \binom{m^2}{m}$ and $p_i = q_i - q_{i-1}$. Let us calculate $E(C_{on})$. For a fixed value of i_1 the on-line cost is at least $i_1 + m$.

$$
\begin{aligned}
E(C_{on}) &\ge \sum_{i=1}^{m} p_i(i+m) = m\sum_{i=1}^{m} p_i + \sum_{i=1}^{m} i(q_i - q_{i-1}) \\
&= m + \sum_{i=1}^{m} iq_i - \sum_{i=1}^{m} i(q_{i-1}) = m + mq_m + \sum_{i=1}^{m-1} iq_i - \sum_{i=0}^{m-1} (i+1)q_i \\
&= 2m - (\sum_{i=1}^{m-1} q_i) - q_0 = 2m - \sum_{i=1}^{m-1} q_i
\end{aligned}
$$

(Since $\sum_{i=1}^{m} p_i = 1$, $q_m = 1$ and $q_0 = 0$.) Let us bound the values q_i:

$$
q_i = \frac{\binom{mi}{m}}{\binom{m^2}{m}} = \frac{(mi)!(m^2-m)!}{(m^2)!(mi-m)!} \le (\frac{i}{m})^m \le e^{i-m}
$$

Thus

$$
\sum_{i=0}^{m-1} q_i \le \frac{e(e^{m-1}-1)}{e^m(e-1)} \le \frac{1}{e-1}
$$

and $E(C_{on}) >= 2m - 1/(e-1)$. The optimal off-line algorithm would assign all jobs in J at time 0, and get $C_{opt} = m+1$. Thus the competitive ratio is $2 - O(1/m)$.

We combine the previous lower bound and the deterministic lower bound with preemption to get the following lower bound:

Theorem 6. *The competitive ratio of any randomized algorithm with preemption is at least $2 - \frac{2}{m+1}$. This is true even if all jobs are unit jobs,*

Proof. For an integer k, $N = (m^k+1)(m-1)+1$ jobs arrive. Denote $L = N/m = m^k + 1 - m^{k-1}$. More m^k jobs: $j_1, ..., j_{m^k}$ arrive so that j_1 depends on a subset J of m of the N jobs, which is chosen uniformly at random among all $\binom{N}{m}$ possible subsets, and for $1 \le i \le m^k - 1$, j_{i+1} depends on j_i and each job in the new sequence m jobs depends on the previous one.

Even with preemption, since each job requires one processing unit, and jobs cannot run simultaneously, at most im jobs can finish before time $i+1$. Let $b_1, ..., b_N$ be the set of the first N jobs sorted according to their finishing time (b_1 finishes first). For $1 \le i \le L$, let J_i be the set $b_{im-m+1}, ..., b_{im}$. Each job in

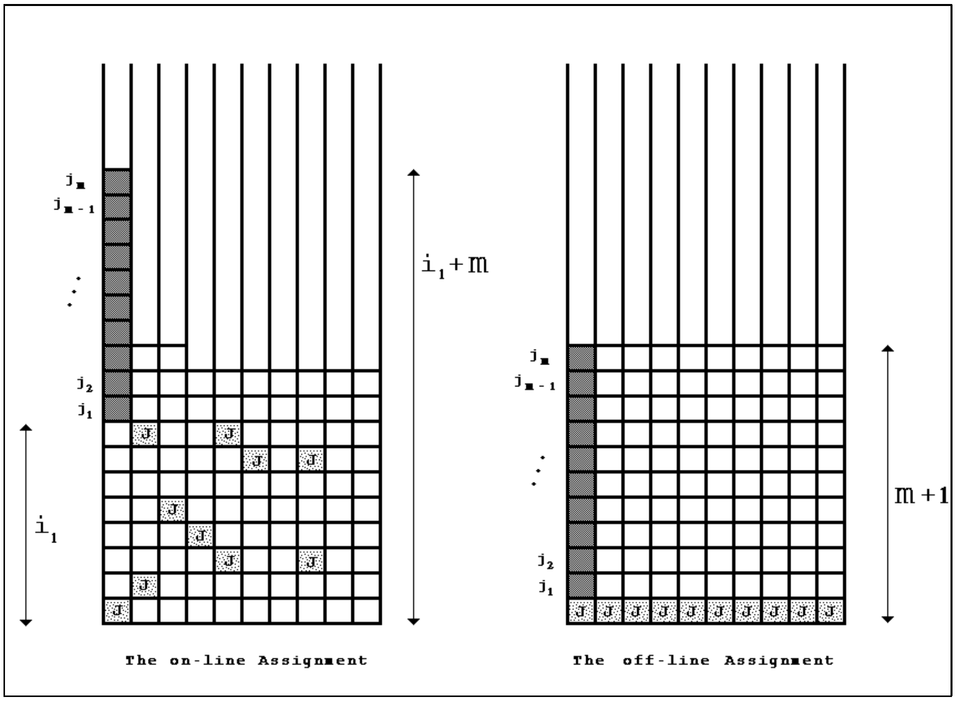

Fig. 2. The on-line and off-line assignments for the sequence in Theorem 5

a set J_i cannot finish before time i. Let I be the set of indices i such that there is at least one job of J in J_i. Let i_1 be the maximum index in I. We define p_i to be the probability that $i = i_1$ for each $1 \le i \le L$. For $0 \le i \le L$, let q_i be the probability that all m jobs of J are chosen among the jobs in the sets $J_1, ..., J_i$ then $q_i = \binom{mi}{m} / \binom{N}{m}$ and $p_i = q_i - q_{i-1}$. Let us calculate $E(C_{on})$. For a fixed value of i_1 the on-line cost is at least $i_1 + m^k$.

$$E(C_{on}) \geq \sum_{i=1}^{L} p_i(i + m^k) = m^k \sum_{i=1}^{L} p_i + \sum_{i=1}^{L} i(q_i - q_{i-1})$$

$$= m^k + \sum_{i=1}^{L} iq_i - \sum_{i=1}^{L} i(q_{i-1}) = m^k + Lq_L + \sum_{i=1}^{L-1} iq_i - \sum_{i=0}^{L-1} (i+1)q_i$$

$$= m^k + L - (\sum_{i=1}^{L-1} q_i) - q_0 = m^k + L - \sum_{i=1}^{L-1} q_i$$

(Since $\sum_{i=1}^{L} p_i = 1$, $q_L = 1$ and $q_0 = 0$.) Let us bound the values q_i:

$$q_i = \frac{\binom{mi}{m}}{\binom{N}{m}} = \frac{(mi)!(N-m)!}{(N)!(mi-m)!} \leq (\frac{i}{L})^m$$

We use the following inequality of Bernoulli:

$$L^{m+1} \geq (L-1)^{m+1} + (m+1)(L-1)^m$$

By induction we can get $L^{m+1} \geq \sum_{i=1}^{L-1}(m+1)i^m$ hence $\sum_{i=1}^{L-1} q_i \leq \frac{L}{m+1}$.

Thus

$$E(C_{on}) \geq m^k + L - L/(m+1) = m^k + (1-1/(m+1))(m^k - m^{k-1}+1) > \frac{2m^{k+1}}{m+1}$$

The optimal off-line algorithm would assign all jobs in J at time 0, and get $C_{opt} = m^k + 1$. Thus the competitive ratio is at least $(2m^{k+1})/((m+1)(m^k+1))$. Since $m^k/(m^{k+1}+1) \to 1$ when $k \to \infty$, The competitive ratio is at least $2m/(m+1) = 2 - 2/(m+1)$

Acknowledgments: I would like to thank Yossi Azar and Jiří Sgall for helpful discussions.

References

1. B. Chen and A. Vestjens. Scheduling on identical machines: How good is lpt in an on-line setting? *To appear in Oper. Res. Lett.*
2. Fabian A. Chudak, David B. Shmoys. Approximation algorithms for precedence constrained scheduling problems on parallel machines that run at different speeds. *Proc. of the 8th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 581-590, 1997.
3. E. Davis and J. M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *J. ACM.* 28(4):721-736, 1981.
4. A. Feldmann, M.-Y. Kao, J. Sgall and S.-H. Teng. Optimal online scheduling of parallel jobs with dependencies. *Proc. of the 25th Ann. ACM symp. on Theory of Computing*, pages 642-651.
5. A. Feldmann, B. Maggs, J. Sgall, D. D. Sleator and A. Tomkins. Competitive analysis of call admission algorithms that allow delay. Technical Report CMU-CS-95-102, Carnegie-Mellon University, Pittsburgh, PA, U.S.A., 1995.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman, New-York, 1979.
7. T. Gonzalez and D. B. Johnson. A new algorithm for preemptive scheduling of trees. *J. Assoc. Comput. Mach.*, 27:287-312, 1980.
8. R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
9. R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math*, 17:263–269, 1969.

10. K. S. Hong and J. Y.-T. Leung. On-line scheduling of real-time tasks. *IEEE Transactions on Computers*, 41(10):1326-1331, 1992.
11. Jeffrey M. Jaffe Efficient scheduling of tasks without full use of processor resources. *The. Computer Science*, 12:1-17, 1980.
12. J.W.S. Liu and C.L. Liu. Bounds on scheduling algorithms for heterogeneous computing systems. *Information Processing 74, North Holland*, 349-353,1974.
13. S. Sahni and Y. Cho. Nearly on line scheduling of a uniform processor system with release times. *Siam J. Comput.* 8(2):275-285, 1979.
14. J. Sgall. On-Line Scheduling - A Survey 1997
15. D. B. Shmoys, J. Wein and D. P. Williamson. Scheduling parallel machines on line. *Siam J. of Computing*, 24:1313-1331, 1995.
16. A. P. A. Vestjens. Scheduling uniform machines on-line requires nondecreasing speed ratios. Technical Report Memorandum COSOR 94-35, Eindhoven University of Technology, 1994. To appear in *Math. Programming.*
17. A. P. A. Vestjens. On-line Machine Scheduling. Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1997.