Multiprocessor Randomized On-line Scheduling

Tomáš Tichý*

August 6, 2002

Abstract

This paper studies randomized on-line non-preemptive scheduling in multiprocessor systems. In this problem each task is specified by its processing time and scheduled on any of m identical processors. The objective is to minimize the expected makespan. We prove lemmas and theorems describing σ_m-competitive randomized algorithms on m processors. The main result is an improvement on the lower bound on the competitive ratio for 3 processors: we show the competitive ratio of any randomized algorithm must be strictly greater than 27/19.

1 Introduction

We study the problem of on-line scheduling of non-preemptive tasks on m identical processors. Each task is specified by its position in the input sequence and by the processing time. The objective is to minimize the expected makespan (length of the schedule).

A randomized on-line algorithm schedules arriving jobs on m processors. When a job arrives, the algorithm decides which processor will be processing the job. This decision can not be changed later and depends only on jobs which arrived before the current job and on the sequence of biased coin flips.

Chen, van Vliet and Woeginger [1] and independently Sgall [3], proved a simple general lower bound σ_m on the competitive ratio for the randomized on-line scheduling on m processors. The optimality of this lower bound was reasonable, because this lower bound $\sigma_2 = 4/3$ is for 2 processors optimal and this is optimal bound in the model with preemptive tasks, see [2]. This work disproves the optimality of the lower bound $\sigma_3 = 27/19$ for 3 processors.

$$\sigma_m = 1 + \frac{1}{\left(\frac{m}{m-1}\right)^m - 1} \tag{1}$$

Further we assume the problem with m processors and that there exists an σ_m-competitive algorithm for contradiction.

*Mathematical Institute, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic. Email tichy@math.cas.cz. Partially supported by Institute for Theoretical Computer Science, Prague (project LN00A056 of MŠMT ČR), grant 201/01/1195 of GA ČR, Partially supported by cooperative grant KONTAKT-ME476/CCR-9988360-001 from MŠMT ČR and NSF.

In section 3 we define a critical job sequence and we study such an algorithm on the sequence. In section 4 we show lemmas and theorems describing behaviour of the algorithm. Their simpler versions for 3 processors are presented in [6].

In section 5 we formulate conditions for $m = 3$ processors which the algorithm has to satisfy for non-negative numbers as a set of equalities and we show that it is not possible. Hence we have a contradiction, there is no $\sigma_3 = 27/19$-competitive algorithm for 3 processors.

2 Preliminaries

- The considered algorithms are the most general randomized algorithms. Since we do not restrict the number of random bits, in principle it may be infinite. However, at each step there are only finitely many configurations, it is equivalent to use a finite number of biased coin flips for scheduling each job.
- Let $load(i)$ denote the ith smallest load of a processor. The load of a processor is the sum of the processing times of all jobs scheduled on it. For a fixed algorithm and an input sequence, $load(i)$ is a random variable. The expected value of $load(i)$ is denoted $Eload(i)$.
- Let *makespan* denote the length of the schedule; it is equal to the maximal load $load(m)$. Let *Emakespan* denote the expected value of the length of the schedule; it is equal to $Eload(m)$.
- Let OPT denote the makespan of the minimal schedule on the input sequence.
- The competitive ratio for the algorithm is the minimum of $Emakespan(P)/OPT(P)$ over all input sequences P.

3 Input sequence

A job sequence is denoted P; P_j denotes the processing time of the jth job; P_{-j} denotes the processing time of the jth job in the reversed ordering of the jobs; P^{-j} denotes the prefix sequence of P finished by the job P_{-j}; $sum(P)$ denotes the sum of all the processing times of all the jobs in the sequence P.

Let us call P_j the *critical job* in a sequence P if and only if

$$P_j = \frac{1}{m-1} \cdot \sum_{i=1}^{j-1} P_i. \tag{2}$$

Our *critical sequence* with parameter q consists of the following jobs:

- $(m-2)(m-1)^{q+m-1}$ jobs, each with the processing time $m^{-q}(m^m - (m-1)^m)$,
- critical jobs $K_1, \ldots, K_q$
  with the processing times $K_i = (m-2)(m-1)^{m-2}(m^m - (m-1)^m)\left(\frac{m}{m-1}\right)^{i-1-q}$,

- an extra job with processing time $(m-1)^{m-1}(m^m-(m-1)^m)$, the processing time is equal to the sum of all previous jobs over $(m-2)$,
- m critical jobs $L_1, \dots, L_m$ with the processing times $L_i = m^{i-1}(m-1)^{m-i}(m^m-(m-1)^m)$.

Let us call *unit jobs* the jobs with processing time $m^{-q}(m^m-(m-1)^m)$, because all jobs in the sequence are integral multiples of this value.

Let us call the *prefix sequence* with parameters q and w, $m \le w \le q$, a sequence with the following jobs:

- $(m-2)(m-1)^{q+m-1}$ jobs, each with the processing time $m^{-q}(m^m-(m-1)^m)$,
- critical jobs $K_1, \dots, K_w$ with the processing times $K_i = (m-2)(m-1)^{m-2}(m^m-(m-1)^m)\left(\frac{m}{m-1}\right)^{i-1-q}$.

The prefix sequence is a prefix of the critical sequence and contains some unit jobs and w critical jobs only.

4 Helpful lemmas and theorems

Lemma 4.1 *Consider any* critical sequence *or* prefix sequence P. *The makespan and all loads in any optimal schedule are equal to* $\frac{1}{m}\mathit{sum}(P)$.

Proof: The last job in the sequence P is a critical job, the definition of the critical job implies $P_{-1} = \frac{1}{m}\mathit{sum}(P)$. This job is scheduled on a processor alone for the whole time, because its processing time is equal to the claimed makespan. Hence we need to prove that there exists a schedule of the remaining jobs P^{-2} on $(m-1)$ processors with the same makespan.

The remaining jobs P^{-2} are scheduled by the greedy algorithm from the largest one to the smallest one (reversed order). Each job is scheduled on a processor with the smallest load. The total time available on the two processors below the requested bound is $\mathit{sum}(P^{-2})$. Therefore there remain at least $\sum_{j=1}^{i} P_j$ of time before scheduling of P_i. The job P_i is either a unit job or a critical job or an extra job. The processing time is always an integral multiple of the processing time of the unit job, hence the unit job can be scheduled. There remains two cases: (i) P_i is a critical job, its processing time is equal to $\frac{1}{m}\sum_{j=1}^{i} P_j$ (because $P_i = \frac{1}{m-1}\sum_{j=1}^{i-1} P_j$), (ii) P_i is an extra job, its processing time is equal to $\frac{1}{m-1}\sum_{j=1}^{i} P_j$ (because $P_i = \frac{1}{m-2}\sum_{j=1}^{i-1} P_j$). Pigeon hole principle implies that there is a processor with at least $\frac{1}{m-1}\sum_{j=1}^{i} P_j$ of time below the claimed bound. Hence we schedule the job P_i on the processor. $\Box$

Lemma 4.2 *Consider a sequence Q extending an arbitrary sequence P by any job and a fixed sequence of biased coin flips. Then for all $1 \le i < m$*

$$\mathit{load}_Q(i) \quad \le \quad \mathit{load}_P(i+1). \tag{3}$$

Proof: Consider the schedule of the sequence P. The algorithm schedules the added job on any processor and we obtain new schedule. This step changes the load of exactly one processor. Consider a fixed i. We assume $load_Q(i) > load_P(i+1)$ for contradiction. Hence there are two processors with loads $load_Q(i)$ and $load_Q(i+1)$ in the new schedule which are strictly greater than load $load_P(i+1)$ in the original schedule. We found a contradiction because two processors were changed, hence $load_Q(i) \leq load_P(i+1)$. $\square$

Lemma 4.3 *Consider a fixed critical or prefix sequence P. The ratios of the expected loads are for all $1 \leq i < m$*

$$\frac{Eload(i+1)}{Eload(i)} = \frac{m}{m-1} \tag{4}$$

and the expected loads are for $1 \leq i \leq m$

$$Eload(i) \quad = \quad Emakespan(P^{-(m+1-i)}) \quad = \quad \frac{\sigma_m}{m} sum(P) \left(\frac{m-1}{m}\right)^{m-i}. \tag{5}$$

Proof: The sum of all loads is equal to the sum of all jobs for a fixed sequence of biased coin flips. Thus this is satisfied also in the expected version:

$$\sum_{i=1}^{m} Eload(i) = sum(P). \tag{6}$$

Recall that P^{-2} is P without the last job and P^{-3} is P without two last jobs, etc. Lemma 4.2 and the definition of the makespan imply that for all $1 \leq i \leq m$

$$load_P(i) \quad = \quad makespan(P^{-(m+1-i)}).$$

and by averaging we obtain the expected versions for $1 \leq i \leq m$

$$Eload_P(i) \quad = \quad Emakespan(P^{-(m+1-i)}). \tag{7}$$

Now we bound the expected makespan. Note that the last m jobs are the critical jobs (for the critical or prefix sequence). The definition of the competitive ratio and Lemma 4.1 imply for all $1 \leq j \leq m$

$$Emakespan(P^{-j}) \quad \leq \quad \sigma_m \cdot OPT(P^{-j}) \quad = \quad \sigma_m \cdot P_{-j}. \tag{8}$$

The definition of the critical jobs implies that $P_{-j} = \frac{1}{m} sum(P) \left(\frac{m-1}{m}\right)^{j-1}$ for all $1 \leq j \leq m$. By this fact and by (7) we obtain the next expression

$$\begin{aligned}
sum(P) &= \sum_{i=1}^{m} Eload_P(i) \\
&\le \sum_{i=1}^{m} Emakespan(P^{-(m+1-i)}) \\
&\le \sum_{j=1}^{m} Emakespan(P^{-j}) \\
&\le \sum_{j=1}^{m} \sigma_m \cdot OPT(P^{-j}) \\
&= \sum_{j=1}^{m} \sigma_m \cdot P_{-j} \\
&= sum(P).
\end{aligned}$$

The left-hand side and the right-hand side are equal, hence all these inequalities are equalities. Therefore

$$\begin{aligned}
\sum_{i=1}^{m} Eload_P(i) &= \sum_{i=1}^{m} Emakespan(P^{-(m+1-i)}) \\
&= \sum_{i=1}^{m} \sigma_m \cdot P_{-(m+1-i)}.
\end{aligned}$$

The equations (9) and the inequalities (7) and (8) imply for all $1 \le i \le m$

$$\begin{aligned}
Eload_P(i) &= Emakespan(P^{-(m+1-i)}) = \sigma_m \cdot P_{-(m+1-i)} \\
&= \frac{\sigma_m}{m} \cdot sum(P) \cdot \left(\frac{m-1}{m}\right)^{m-i},
\end{aligned} \tag{9}$$

hence the equalities (5) in the statement are satisfied. The fixed ratios (4) follow from the explicit expressions for the expected loads (5). □

Lemma 4.4 *Consider a prefix sequence P and a prefix sequence Q extending the sequence P by one critical job ($P = Q^{-2}$). Then for all $1 \le i < m$*

$$\begin{aligned}
Eload_Q(i) &= Eload_P(i+1) \quad \textit{and} \\
Eload_Q(m) &= Eload_P(1) + Q_{-1}.
\end{aligned}$$

Proof: The sequence Q extends P by a critical job, therefore $sum(Q) = \frac{m}{m-1} sum(P)$. Hence the expressions for $Eload_Q(i)$ and $Eload_P(i+1)$ in Lemma 4.3 are equal for all $1 \le i < m$. The facts $sum(Q) = sum(P) + Q_{-1}$, $sum(Q) = \sum_{i=1}^{m} Eload_Q(i)$, and $sum(P) = \sum_{i=1}^{m} Eload_P(i)$ imply $Eload_Q(m) = Eload_P(1) + Q_{-1}$. □

Lemma 4.5 *Consider a prefix sequence P and a prefix sequence Q extending the sequence P by one critical job ($P = Q^{-2}$). The algorithm deterministically schedules the job on a processor with the smallest load.*

Formally, for each sequence of biased coin flips, for all $1 \le i < m$:

$$\begin{aligned} load_Q(i) &= load_P(i+1) \quad and \\ load_Q(m) &= load_P(1) + Q_{-1}. \end{aligned}$$

Note: *This does not mean that the algorithm is deterministic. The algorithm behaves deterministically on any critical job which extends a prefix sequence.*

Proof: The last equality $load_Q(m) = load_P(1) + Q_{-1}$ follows from the other equalities, because $\sum_{i=1}^{m} load_Q(i) = sum(Q)$.

Lemma 4.2 implies inequality for all $1 \le i < m$ $load_Q(i) \le load_P(i+1)$. The statement requires us to prove the corresponding equality for all $1 \le i < m$ $load_Q(i) = load_P(i+1)$. Consider a fixed $1 \le i < m$ and let us assume for contradiction that

$$load_Q(i) < load_P(i+1) \tag{10}$$

Then the probability of satisfying (10) is positive, because there are finitely many possible configurations, hence the probability of each reachable configuration is positive. The positive probability of (10) implies $Eload_Q(i) < Eload_P(i+1)$ by the definition of the expected value and Lemma 4.2. But this is in contradiction with Lemma 4.4 which gives $Eload_Q(i) = Eload_P(i+1)$. Hence the lemma is proven. □

Theorem 4.6 *Consider a fixed prefix sequence Q with parameters $q = w$ and a fixed σ_m-competitive algorithm on m processors. The loads are proportional to $1 : \ldots \left(\frac{m}{m-1}\right)^i \ldots : \left(\frac{m}{m-1}\right)^{m-1}$ for q tending to infinity. For an arbitrarily small $\varepsilon > 0$ and a sufficiently large q (number of the critical jobs), for any sequence of biased coin flips, for all $1 \le i \le m$*

$$\left| load(i) - (m-1)^{m-i} m^{i-1} \cdot (m-1)^{m-1}(m-2) \right| \quad < \quad \varepsilon. \tag{11}$$

Proof: Let us assume $q = m \cdot k$. Let us denote P the prefix sequence with parameters q and $w = m$. The sequence Q extends P by $q - m$ critical jobs. The algorithm schedules each of the $q - m$ critical jobs on the smallest processor by Lemma 4.5. Let us call the processors A, B, C, ..., the first job is scheduled on A, the second on B, the third on C, ..., the $(m+1)$th on A, etc. There are $k-1$ periods in the scheduling. If jobs Q_j and Q_{j+1} are the critical jobs, then $Q_{j+1} = \frac{m}{m-1} Q_j$ by the definition of the critical job. These facts imply for all $1 \le i \le 3$

$$load_Q(i) = load_P(i) + \frac{sum(P)}{m-1} \cdot \sum_{j=0}^{k-2} \left(\frac{m}{m-1} \right)^{i-1+mj}. \tag{12}$$

The definition of the load implies $load_P(i) \le sum(P)$ for all $1 \le i \le m$. There are $(m-2)(m-1)^{q+m-1}$ unit jobs, each with the processing time $m^{-q}(m^m - (m-1)^m)$, m critical jobs for $j = 1, \ldots, m$, the processing time of the jth critical job is $(m-2)(m-1)^{q+m-1}m^{-q}(m^m-(m-1)^m)\frac{1}{m-1}\left(\frac{m}{m-1}\right)^{j-1}$. Hence $sum(P) = (m-2)(m-1)^{q+m-1}m^{-q}(m^m-(m-1)^m)\cdot\left(1+\sum_{j=1}^{m}\frac{1}{m-1}\left(\frac{m}{m-1}\right)^{j-1}\right) = (m-2)(m-1)^{q+m-1}m^{-q}(m^m-(m-1)^m)\left(\frac{m}{m-1}\right)^{m}$. We sum the geometric sequence on the right-hand side of (12). We obtain equality for all $1 \le i \le m$ $load_Q(i) = (m-2)(m-1)^{m-1}m^{i-1}(m-1)^{m-i}$ for q tending to infinity. Note that the sufficient q to satisfy (11) for an arbitrary $\varepsilon > 0$ is $\Omega(-\log\varepsilon)$. □

The motivation of the next theorem is to show that the m critical jobs at the end of a critical sequence are usually scheduled on different processors, except for some degenerate cases. Namely, if the loads differ in each step of scheduling then the jobs are scheduled on different processors. Otherwise some loads are equal and the theorem may give no information.

Theorem 4.7 *Consider a fixed sequence of biased coin flips, a σ_m-competitive algorithm, a critical sequence P. Then for all $1 \le i \le m$*

$$load_P(i) \quad = \quad makespan(P^{-(m+1-i)}). \tag{13}$$

Proof: Lemma 4.2 implies that for all $1 \le i \le m$

$$load_P(i) \quad \le \quad makespan(P^{-(m+1-i)}).$$

Lemma 4.3 implies

$$Eload_P(i) \quad = \quad Emakespan(P^{-(m+1-i)}).$$

We use the same argument which we use in the proof of Lemma 4.5: the probability of each reachable configuration is positive, hence the equality in the statement of the theorem is proven. □

5 Main result

The main result is for $m = 3$ processors only. Therefore, further we assume that $m = 3$ and a fixed $\sigma_3 = 27/19$-competitive randomized algorithm for contradiction.

5.1 Idea

The behavior of the algorithm on a critical sequence P with parameter q is studied. The sequence P consists of the prefix sequence P^{-5}, with many critical jobs at the end, and one extra job with processing time 76 and 3 critical jobs with processing times 76, 114, 171. The loads on the prefix sequence P^{-5} are 16, 24, 36 for ε tending to zero. Some linear inequalities and linear equations follow from the previous theorems.

Consider arbitrarily small $\varepsilon > 0$, choose a prefix sequence P^{-5} with q critical jobs at the end by Theorem 4.6. Linear conditions depending on ε follow from the definition of the expected load and by Theorem 4.6. Consider the conditions for $\varepsilon = 0$, we have a linear program. If the linear program is infeasible then there is $\varepsilon > 0$ such that the conditions are also infeasible by the theory of linear programming.

5.2 Conditions

There are 3^4 possibilities of how to schedule the last 4 jobs of P. Each possibility can be represented as a 4-tuple of A, B, C. Let us denote processors A, B, C in the ascending ordering by their loads after scheduling P^{-5}. The ith job (P^{-5+i}) will be scheduled on a processor at the ith place in the 4-tuple.

Consider any fixed sequence of jobs for a fixed choice of $\varepsilon > 0$. The algorithm is randomized, let p_c denote the probability that the algorithm schedules the last 4 jobs according to the 4-tuple c. The sum of these probabilities is 1, which gives the first condition.

$$\sum_{c \in \{1,2,3\}^4} p_c = 1 \tag{14}$$

Let us call two values ε-close if they differ less then ε. Consider the 4-tuple c. The last 4 jobs of P are scheduled according to c on the loads 16, 24 and 36, then we denote L_c^i the ith smallest load. The loads on P^{-5} are ε-close to 16, 24, 36 by Theorem 4.6, hence the loads on P, where the last 4 jobs are scheduled according to c, are ε-close to L_c^i.

The last 4 jobs of P are 76, 76, 114 and 171 by the definition of P. Hence by Theorem 4.6 the expected loads of the sequence P are ε-close to 108, 162 and 243, then

$$\begin{aligned} 108 - \varepsilon \;\leq\; & \sum_{c \in \{A,B,C\}^4} L_c^1 \cdot p_c \;\leq\; 108 + \varepsilon \quad \text{and} \\ 162 - \varepsilon \;\leq\; & \sum_{c \in \{A,B,C\}^4} L_c^2 \cdot p_c \;\leq\; 162 + \varepsilon. \end{aligned} \tag{15}$$

By the limit version of the expressions (15) on ε tending to zero we obtain better forms of the conditions.

$$\begin{aligned} \sum_{c \in \{A,B,C\}^4} L_c^1 \cdot p_c &= 108 \quad \text{and} \\ \sum_{c \in \{A,B,C\}^4} L_c^2 \cdot p_c &= 162. \end{aligned} \tag{16}$$

Note that the analogous condition for L_c^3 is not necessary to prove the theorem, as it linearly depends on these two conditions.

It is possible to consider the limit version only to prove the infeasibility. If the system (14), (16) is infeasible then exists sufficiently small neighbourhood ($\varepsilon > 0$) of the right-hand sides of the system in which the feasibility does not change.

The lower bound sequence P is a critical sequence. Hence by Theorem 4.7 it can be easily checked that all 4-tuples except ABCA, ABAC, ACBA, BACA, BAAC, BCAB, CABA and CBAB are impossible (each has zero probability $p_c = 0$).

Example: Remember that the starting schedule is $16, 24, 36$ and the last four jobs are $76, 76, 114, 171$. How to compute the second column? Its 4-tuple is ABAC. Hence the first job 76 is scheduled on

16 (denoted A), the second job 76 is scheduled on 24 (denoted B), the third job 114 is scheduled on $16 + 76$ (denoted A) and the fourth job 171 is scheduled on 36 (denoted C). Hence the loads are $16 + 76 + 114 = 206$ on A, $24 + 76 = 100$ on B and $36 + 171 = 207$ on C. The obtained schedule is $100, 206, 207$, first two values $100, 206$ are coefficients in the conditions. Note that the 4-tuple satisfies Theorem 4.7, because the makespans (through the scheduling the four jobs) are 92, 100, 206, 207 and the last 3 are equal to the final loads.

5.3 Conditions

We consider the conditions (14) and (16) with the following variables p_c for c in ABCA, ABAC, ACBA, BACA, BAAC, BCAB, CABA and CBAB, other variables are equal to zero. The 4-tuples in each of the following couples give the same coefficients in the conditions, the couples are: ABCA and BACA, ABAC and BAAC, ACBA and CABA, BCAB and CBAB. Hence we denote $a = p_{ABCA} + p_{BACA}$, $b = p_{ABAC} + p_{BAAC}$, $c = p_{ACBA} + p_{CABA}$ and $d = p_{BCAB} + p_{CBAB}$. Hence we formulate the system of conditions

$$\begin{array}{ccccccccc} a & + & b & + & c & + & d & = & 1, \\ 100 \cdot a & + & 100 \cdot b & + & 112 \cdot c & + & 112 \cdot d & = & 108, \\ 150 \cdot a & + & 206 \cdot b & + & 138 \cdot c & + & 130 \cdot d & = & 162. \end{array} \qquad \text{and}$$

The solution depending on d of the system is

$$a = -\tfrac{d}{7} - \tfrac{1}{42} \quad , \quad b = \tfrac{d}{7} + \tfrac{5}{14} \quad , \quad c = -d + \tfrac{2}{3} \ .$$

The variables a and d are probabilities, hence $a, d \geq 0$, but $d \geq 0$ implies $a \leq -\frac{1}{42}$. Hence a contradiction is found and the conditions can not be satisfied. The next theorem follows:

Theorem 5.1 *Each randomized on-line algorithm for scheduling on* 3 *processors has a competitive ratio greater than* $\sigma_3 = 27/19$.

6 Concluding notes

It was reasonable to believe that the σ_m lower bound on competitive ratio for m processors is optimal. There are two main reasons: the optimality on 2 processors and the optimality of the lower bound in the model with preemptive tasks. This paper disproves the optimality on 3 only. There is no more reason to believe the optimality for $m > 3$. The effort to generalize the proof for other $m > 3$ was not successful. All tried generalizations of the used sequence gave the known σ_m lower bound for m processors. However this result does not relate to the possible optimality of the asymptotical lower bound on the competitive ratio $\frac{e}{e-1}$.

This paper disproves optimality only. The linear program can be formulated in a weaker version and we can obtain a contradicting competitive ratio. But the theorems used are strongly dependent on the assumption of an exactly σ_3-competitive algorithm. Hence such a number is not a new lower bound.

Seiden [5] introduced two randomized on-line algorithms for the problem of randomized on-line scheduling of non-preemptive tasks. These algorithms always schedule a job on two smallest processors. These algorithms are much simpler than the general algorithms, but still hard to analyze. We consider a class of algorithms which schedule a job on a processor with the smallest load or on a processor with the ith smallest load, where i is fixed for the algorithm. The upper bounds on the competitive ratio of Seiden's algorithms are very distant from the σ_m lower bounds. We study the lower bounds on the competitive ratio over all such algorithms in [7]. We constructed a computer proof for $m = 3, 5, 6$ of lower bounds 1.42533, 1.49509, 1.50477. The bounds are only slightly better than the σ_m lower bounds 1.42105, 1.48738, 1.50352.

References

[1] B. Chen, A. van Vliet, G. Woeginger: *A Better Lower Bound for Randomized On-line Scheduling Algorithms.* Information Processing Letters 51 (5):219-222, 1994

[2] B. Chen, A. van Vliet, G. Woeginger: *An Optimal Algorithm for Preemptive On-line Scheduling.* Operation Research Letters 18 127-131, 1995

[3] J. Sgall: *A Lower Bound for Randomized On-Line Multiprocessor Scheduling.* Information Processing Letters 63 (1):51-55, 1997

[4] A. Fiat, G. J. Woeginger (Eds.): *Online Algorithms - The State of the Art.* Springer, Berlin, 1998

[5] S. Seiden: *Randomized Algorithms for that Ancient Scheduling Problem.* Proceedings of the 5th Workshop on Algorithms and Data Structures, LNCS 1272 210-223, 1997

[6] T. Tichý: *Randomized On-line Scheduling on 3 processors.* Operation Research Letters, 2002

[7] T. Tichý: *A Lower Bound for Restricted On-line Algorithms for Scheduling.* Week for Doctoral Students'2002 Proceedings, Faculty of Mathematics and Physics, Charles University, Prague, 2002