Randomized algorithms for on-line scheduling problems: how low can't you go?
Stougie, L.; Vestjens, A.P.A.

Published: 01/01/2000

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Randomized algorithms for on-line scheduling problems: How low can't you go?

L. Stougie
A.P.A. Vestjens

/ department of mathematics and computing science
Randomized algorithms for on-line scheduling problems:
How low can’t you go?

Leen Stougie
Department of Mathematics and Computing Science
Technische Universiteit Eindhoven
Centre for Mathematics and Computer Science (CWI)
Amsterdam

Arjen P.A. Vestjens
Centre for Quantitative Methods-CQM

June 29, 2000

Abstract

We prove lower bounds on the competitive ratio of randomized algorithms for several on-line scheduling problems. The main result is a bound of $e/(e-1)$ for the on-line problem with objective minimizing the sum of completion times of jobs that arrive over time at their release times and are to be processed on a single machine. This lower bound shows that a randomized algorithm designed in [Chekuri et al. 1997] is a best possible randomized algorithm for this problem.

Keywords and -Phrases: on-line scheduling, competitive analysis, randomized algorithm.

1 Introduction

This paper concerns on-line versions of scheduling problems in which jobs become known only at their release time. The jobs are indexed by non-decreasing release times. At the release time also the processing time becomes known. In [Feuerstein et al. 1999] this on-line paradigm is called the REAL-TIME-model. At any point in time $t$ there is only information available about jobs that have been released up to $t$. Nothing is known about future jobs, neither if there will be any future jobs. Based on the information up to time $t$ the schedule up to $t$ must have been decided upon, history is irrevocable. We will refer to the problem with a priori complete information as the off-line problem. A beautiful survey of on-line scheduling problems is given in [Sgall1998].

The performance behaviour of algorithms for on-line problems is measured through competitive analysis (see [Borodin and El-Yaniv 1998]). The outcome of an algorithm for an on-line problem is compared to that of an adversary, who, knowing the algorithm, provides the sequence of requests and schedules them himself in an optimal -off-line- way. An algorithm is said to be $\rho$-competitive for an on-line problem if for any instance of the problem its solution value is no more than $\rho$ times the optimal off-line solution value (possibly plus some constant).

Usually the lack of information makes that the optimal off-line solution can not be obtained by any algorithm on every input sequence. There exists a lower bound on the competitive ratio of any algorithm for the on-line problem. We call an algorithm best possible if its competitive ratio is equal to the lower bound. For many problems there are no matching lower and upper bounds known.

In case the adversary plays against a deterministic algorithm he knows exactly the algorithm's behaviour, thus being a very powerful adversary. Here we consider randomized algorithms and measure their competitive ratio by playing against an adversary that is called oblivious in the literature (see a.o.
[Borodin and El-Yaniv 1998)]. Such an adversary knows the probability distribution defining the randomized algorithm but none of the outcomes of the random experiments involved are revealed to him. Thus, he will not be able to adapt the input sequence since he is not informed how the algorithm has performed on the input sequence he has presented so far. He selects the optimal off-line solution on the input sequence he provides.

We present a general result in Section 2 that allows us to derive lower bounds on competitive ratios through probabilistic analysis of algorithms on random input sequences. The random instances that we use to establish the various lower bounds are all variations on a common theme. We explain this theme also in Section 2.

The most prominent result, presented in Section 3, is a lower bound of $\frac{e}{e-1}$ on the competitive ratio of randomized algorithms for the on-line scheduling problem of minimizing the sum of completion times, also called the latency, of a sequence of jobs to be processed on a single machine. Within the classification of [Lawler et al. 1993] the off-line problem is denoted by $1|\tau_j|\sum C_j$. It is NP-hard, [Garey and Johnson 1979]. An enormous research effort has recently shown that polynomial time approximation schemes exist for this problem and for its extensions to parallel machines and to weighted completion times (even on uniform machines and on a fixed number of unrelated machines) [Sktella and Woeinger 2000], [Afrati et al. 1999].

Our lower bound proves that a randomized algorithm designed in [Chekuri et al. 1997] for this problem is best possible.

In a recent paper [Seiden 1999] provides a general framework for proving lower bounds for on-line REAL-TIME optimization problems and from there reproves our lower bound. He also reproves some of the other results in this paper and improves the bound for the maximum delivery time problem, and hence for the two machine flow shop problem.

The other results we establish here in Section 4 are:

- A lower bound of 1.302 for the on-line maximum delivery time problem on a single machine and a lower bound of 1.265 for the same problem on identical parallel machines. The first bound is improved to 1.349 in [Seiden 1999].
- A lower bound of $\Omega(\sqrt{n})$ for the on-line flow time problem on a single machine.
- A lower bound of $4 - \sqrt{2}$ for the on-line makespan problem on identical parallel machines.
- A lower bound of 1.302 for the on-line flow shop scheduling problem on two machines minimizing makespan. The bound is improved to 1.349 in [Seiden 1999].
- A lower bound of $5/4$ for the on-line open shop scheduling problem on two machines minimizing makespan.

## 2 Deriving lower bounds

It is general practice in on-line optimization to use Yao’s Minimax Principle to derive lower bounds on competitive ratios of randomized algorithms. Yao’s Minimax Principle has been introduced in [Yao 1977]. Applications of the principle in on-line optimization can be found in [Borodin and El-Yaniv 1998]. However, as pointed out in [Borodin and El-Yaniv 1999], straightforward application of the principle is often not possible. In many on-line problems some subtle care has to be taken, which leads to annoying technical details. Instead we will use here a much more direct argument to get the desired result given in the following lemma.

We introduce some notation first. For an on-line optimization problem, let $\mathcal{I}$ be the class of possible input sequences, and $\mathcal{A}$ be a class of possible deterministic algorithms, both possibly infinite. A random input sequence $I_p$ is defined as a probability distribution $p$ over $\mathcal{I}$, and a randomized algorithm $A_q$ is defined as a probability distribution $q$ over $\mathcal{A}$. Let $Z^A(I)$ and $Z^{OPT}(I)$ denote the solution value
produced by algorithm $A$ and the optimal solution value on instance $I$, respectively. The competitive ratio of a randomized algorithm $A_q$ against an oblivious adversary is defined as

$$\max_{I \in \mathcal{I}} \frac{E[Z^A(I)]}{Z^{OPT}(I)}.$$\nonumber

**Lemma 2.1** Given an on-line optimization problem, with possible input sequences $I$, and possible algorithms $A$, both possibly infinite, for any random sequence $I_p$, and any randomized algorithm $A_q$, we have

$$\min_{\mathcal{A} \in \mathcal{A}} \frac{E_{I_p}[Z^A(I_p)]}{Z^{OPT}(I_p)} \leq \max_{I \in \mathcal{I}} \frac{E_{A_q}[Z^{A_q}(I)]}{Z^{OPT}(I)}$$

provided that the left hand side is bounded.

**Proof.** Suppose that there exists a randomized algorithm $A_q$ with competitive ratio $c$. Then $\forall I \in \mathcal{I}$

$$cZ^{OPT}(I) \geq E_{A_q}[Z^{A_q}(I)].$$

Hence also

$$cE_{I_p}[Z^{OPT}(I_p)] \geq E_{I_p}[E_{A_q}[Z^{A_q}(I_p)]] = E_{A_q}[E_{I_p}[Z^{A_q}(I_p)]] \geq \min_{\mathcal{A} \in \mathcal{A}} E_{I_p}[Z^A(I_p)].$$

Through this lemma, computing strong lower bounds boils down to choosing adversarial random input sequences.

The random input sequences used in the following sections for establishing the lower bounds for the various problems are all variations on a common theme. They all start with one job or a set of jobs with unit processing time. Any on-line algorithm must start processing this job (or set of jobs) at some time $t$. Such an algorithm is denoted by $OL(t)$. Then with some probability another job or set of jobs arrives at some random time. We calculate the expected value of the optimal off-line solution, and the expected value of the solution produced by $OL(t)$. We then choose the best value for $t$, to obtain a lower bound on the ratio between these two expected values. The argument is then finished by application of Lemma 2.1.

In the following sections we derive lower bounds on competitive ratio's of randomized algorithms for various scheduling problems following the above recipe. We will prove in detail the one for the problem of minimizing latency on a single machine. For the other problems we just give the random instance.

### 3 Minimizing latency on a single machine

[Phillips et al. 1995] [Hoogeveen and Vestjens 1996] and [Stougie 1995] (see [Vestjens 1997] for a concise overview) independently devised 2-competitive deterministic algorithms for the on-line scheduling problem of minimizing latency on a single machine while jobs arrive over time. In [Vestjens 1997] it is proved that 2 is best achievable as a competitive ratio for deterministic algorithms for this problem.

In [Chekuri et al. 1997] a randomized algorithm for this on-line scheduling problem is designed that is $\frac{\epsilon - 1}{\epsilon}$-competitive. They also showed that no randomized algorithm can be less than $\frac{1}{3}$-competitive against an oblivious adversary. We prove that, actually, no randomized algorithm can be less than $\frac{\epsilon - 1}{\epsilon}$-competitive, thus settling that the randomized algorithm of [Chekuri et al. 1997] is best possible.
Theorem 3.1 Any randomized algorithm for the on-line single machine latency problem must have competitive ratio greater or equal to $\frac{e}{e-1} \approx 1.582$ against an oblivious adversary.

Proof. Consider the following class of random input sequences depending on $n$.

- At time 0 one job with processing time 1 arrives.
- With probability $1 - \frac{e-1}{n}$ no further jobs arrive.
- With probability $\frac{e-1}{n}$ one set of $n-1$ jobs with processing time 0 arrives at some time $x$, which is a random variable over the interval $(0,1]$ having probability density function $f(x) = e^{-x}$.

First we derive the expected optimal objective value $E[\sum C_j^{opt}]$ on the random instance. Observe that if only the first job is released the optimal value is equal to 1. If at any time $x \leq 1 - \frac{1}{n}$ the set of $n-1$ jobs is also released then it is profitable to schedule the $n-1$ jobs before the first job, yielding objective value $nx + 1$. If the $n-1$ jobs arrive between time $1 - \frac{1}{n}$ and 1 it is better to process the first job first giving a sum of completion times equal to $n$. These observations lead to the following upper bound on the expected optimal solution value.

$$E[\sum C_j^{opt}] \leq \left(1 - \frac{e-1}{n}\right) \cdot 1 + \frac{e-1}{n} \int_0^1 (nx + 1)f(x)dx$$

$$= 1 - \frac{e-1}{n} + \frac{e}{n} \int_0^1 e^{-x}(nx + 1)dx$$

$$= 1 - \frac{e-1}{n} + \frac{e}{n}(-ne^{-1} - ne^{-1} - e^{-1} + n + 1)$$

$$= e - 1.$$

Any deterministic on-line algorithm will have to start scheduling the first job at some point in time. Consider an algorithm that starts processing the first job at time $t$, unless the set of $n-1$ jobs arrives before $t$. In the latter case it is obviously better for the algorithm to schedule the $n-1$ jobs first, before starting the (big) first job. Moreover, since the deterministic algorithm knows the distribution, he will start the first job immediately after having processed the other jobs since no further jobs will arrive. In this case a cost of $nx + 1$ will be attained. In case the set of $n-1$ jobs arrives after $t$ then these jobs have to wait until the first job is finished producing an objective value of $(t+1)n$. Obviously, if the set of $n-1$ jobs does not arrive the only job will be completed at time $t+1$. We will denote the expected solution value of an on-line algorithm that does not start the first job before time $t$ by $E[\sum C_j^{OL(t)}]$. Now,

$$E[\sum C_j^{OL(t)}] = (1 - \frac{e-1}{n})(t+1) + \frac{e-1}{n} \int_0^t \frac{e}{e-1}e^{-x}(nx + 1)dx$$

$$+ \frac{e-1}{n} \int_t^1 \frac{e}{e-1}e^{-x}(t+1)ndx$$

$$= e - \frac{t(e-1) + e^{1-t} - 1}{n}$$

$$\geq e - \frac{e-1}{n}.$$

This last inequality follows from minimizing the expected value with respect to $t$ over the interval $[0,1]$. The minimum is obtained at either $t = 0$ or $t = 1$. Thus, for any $t \in [0,1]$,

$$\frac{E[\sum C_j^{OL(t)}]}{E[\sum C_j^{OPT}]} = \frac{e - \frac{e-1}{n}}{e - 1}.$$

The ratio can be made arbitrarily close to $\frac{e}{e-1}$, by choosing $n$ large enough. This is true for any fixed $t$. The observation that it is useless for any algorithm to start the first job after time 1 shows that the above ratio holds for the best possible deterministic algorithm on this random instance.
That the ratio derived above is indeed a lower bound on the competitive ratio of any randomized algorithm follows from application of Lemma 2.1.

4 Lower bounds for other scheduling problems

In this section we establish lower bounds for some other on-line scheduling problems. Our guess is that these bounds are strong. However, for none of these problems randomized algorithms with matching competitive ratio's have been designed so far.

Minimizing flow time on a single machine

The setting is the same as in the problem of the previous section. The objective is to minimize the sum of completion times of jobs minus their release times. This sum is called the flow time. The corresponding off-line problem \( (1|\pi_j| \sum F_j) \) has the same optimal solution as the latency problem of the previous section, but not the same optimal solution value. This leads to dramatically different approximability properties. In [Kellerer et al. 1999] it is shown that any off-line polynomial time algorithm has a worst-case performance ratio of \( \Omega(n^{1/2-\epsilon}) \), for any \( \epsilon > 0 \), unless \( P = NP \).

For the on-line version of the problem there is an obvious lower bound of \( \Omega(n) \) on the competitive ratio of any deterministic algorithm, which to the best of our knowledge is common knowledge in the field of on-line scheduling, but has never been published. We show that randomization will not be able to bring the competitive ratio down to a constant.

**Theorem 4.1** Any randomized algorithm for the on-line single machine flow time problem has a competitive ratio against an oblivious adversary of \( \Omega(n) \).

**PROOF.** The lower bound is obtained using the following random instance:

- At time 0 a job arrives requiring processing time 1.
- One set of \( n - 1 \) jobs all requiring processing time 0 arrives at some random time \( x \) in the interval \([0, \sqrt{n}]\), with probability density function \( f(x) = e^{1-x/\sqrt{n}}/((e - 1)\sqrt{n}) \).

Minimizing delivery time on a single machine

In this problem a job consists of two different operations. The first is processing on a single machine to manufacture a product and the second is delivering the product. Like processing of the job, also delivering requires a specific time. There is no limit on the number of jobs that can be delivered simultaneously. The objective is to minimize the time at which the last job is delivered. We give the following result only for historical reasons, since very recently Seiden has improved the bound (see Introduction).

**Theorem 4.2** Any randomized algorithm for the on-line maximum delivery time problem on a single machine has a competitive ratio against an oblivious adversary of at least approximately 1.302.

**PROOF.** The lower bound is obtained using the following random instance, where \( a \) is the solution of the equation \( e^{-a} + 2a - 2 = 0 \).

- At time 0 a job with processing requirement 1 and delivery time 0 arrives.
- With probability \( e^{-a} \) no further jobs arrive.
• With probability $1 - e^{-a}$ one job with processing requirement 0 and delivery time 1 arrives at some random time $x$ in the interval $[0, a]$ according to the probability density function $f(x) = \frac{e^{a-x}}{(e^a - 1)}$.

The bound compares with a best possible competitive golden ratio deterministic algorithm for the problem [Hoogeveen and Vestjens 2000].

**Remark 4.1** By the way the random instance for the maximum delivery problem on a single machine is constructed it can also be used for the on-line two-machine flow shop problem minimizing makespan, leading to the same lower bound for this problem.

**Minimizing delivery time on identical parallel machines**

Here we consider the same problem as the previous one but now on identical parallel machines.

**Theorem 4.3** Any randomized algorithm for the on-line maximum delivery time problem on identical parallel machines has a competitive ratio against an oblivious adversary greater or equal to 1.265.

**Proof.** The lower bound is obtained using the following random instance, with $m$ the number of machines and $a$ the solution of the equation $2e^{-a} + 3a - 3 = 0$:

- At time 0 a set of $m$ jobs is presented all having processing time 1 and delivery time 0.
- With probability $e^{-a}$ no further jobs arrive.
- With probability $1 - e^{-a}$ at some random time $x \in [0, a]$, another set of $m$ jobs arrives all requiring some random processing time $x$ and having delivery time 1, where $x$ has probability density function $f(x) = \frac{e^{a-x}}{e^a - 1}$ on the interval $[0, a]$.

**Minimizing makespan on identical parallel machines**

The on-line version of the classical problem of minimizing makespan of a schedule on identical parallel machines is considered here.

**Theorem 4.4** Any randomized algorithm for the on-line makespan problem on identical parallel machines has a competitive ratio against an oblivious adversary greater or equal to $4 - 2\sqrt{2}$.

**Proof.** The lower bound is obtained using the following random instance:

- At time 0 a set of $m$ jobs is presented all having processing time 1.
- With probability $2 - \sqrt{2}$ no further jobs arrive.
- With probability $\sqrt{2} - 1$ one job requiring processing time $3 - \sqrt{2}$ arrives at time $\sqrt{2} - 1$. 

6
Minimizing makespan in a two-machine open shop

Finally, we derive a lower bound for another shop scheduling problem. It concerns the open shop problem on two machines, Machines 1 and 2. Each job consists of two operations. The first operation is to be processed on Machine 1 and the second on Machine 2. The operations can be executed in any order, but cannot be executed simultaneously. The objective is to minimize makespan when jobs arrive over time. The best possible deterministic on-line algorithm for this problem is 3/2-competitive [Vestjens 1997]. If preemption is allowed the best possible deterministic algorithm has competitive ratio 5/4. We show that the latter is also a lower bound for any randomized algorithm for the problem without preemption.

Theorem 4.5 Any randomized algorithm for the on-line two machine open shop problem minimizing makespan has a competitive ratio against an oblivious adversary greater or equal to 5/4.

Proof. The lower bound is obtained using the following random instance:

- At time 0 one job arrives requiring processing time 1 on both machines.
- At time 1 another job arrives. With probability 1/2 this job requires processing time 1 on Machine 1 and 0 on Machine 2. With equal probability, it requires processing time 0 on Machine 1 and 1 on Machine 2.

\[ \Box \]

5 Conclusion

This work shows that for on-line scheduling problems randomization may give improvements in competitive ratio if compared to their deterministic counterparts. We can not be conclusive on this statement since we have not found randomized algorithms for the problems discussed here with good competitive ratio. This remains work for the future. Only for the on-line latency problem on a single machine our lower bound shows that a randomized algorithm in [Chekuri et al. 1997] is best possible and indeed its competitive ratio is lower than the lower bound of 2 on the competitive ratio's of deterministic algorithms for this problem.

Acknowledgements. We thank Jiri Sgall for drawing our attention to Lemma 2.1. We also gratefully acknowledge the people that referred to a manuscript of this paper from 1996, as well as those that asked and are asking for copies. They encouraged us to write this paper eventually and submit it to a journal.

References


