Algorithmica (2002) 32: 36–60
DOI: 10.1007/s00453-001-0073-z

Algorithmica

On-Line Multi-Threaded Paging[1]

E. Feuerstein[2] and A. Strejilevich de Loma[2]

Abstract. In this paper we introduce a generalization of Paging to the case where there are many threads of requests. This models situations in which the requests come from more than one independent source. Hence, apart from deciding *how* to serve a request, at each stage it is necessary to decide *which* request to serve among several possibilities.

Four different on-line problems arise depending on whether we consider fairness restrictions or not, with finite or infinite input sequences. We study all of them, proving lower and upper bounds for the competitiveness of on-line algorithms.

The main competitiveness results presented in this paper state that when no fairness restrictions are imposed it is possible to obtain competitive algorithms for finite and infinite inputs. On the other hand, for the fair case in general there exist no competitive algorithms.

In addition, we consider three definitions of competitiveness for infinite inputs. One of them forces algorithms to behave efficiently at every finite stage, while the other two aim at comparing the algorithms' steady-state performances. A priori, the three definitions seem different. We study them and find, however, that they are essentially equivalent. This suggests that the competitiveness results that we obtain reflect the intrinsic difficulty of the problem and are not a consequence of a too strict definition of competitiveness.

Key Words. Competitive analysis, Fairness, Multi-tasking systems, On-line algorithms, Paging.

1. Introduction. The *Paging* problem consists of managing a two-level memory, one level having limited capacity and fast access (the cache) and the other one having slow access but potentially unlimited capacity. A *Paging algorithm* is given a sequence of page references; at each step the algorithm must ensure that the requested page is in fast memory, perhaps evicting another page to make room for the incoming one. A *page fault* occurs each time a page must be brought into fast memory. The goal of a Paging algorithm is to minimize the total number of page faults over the sequence of requests. An *on-line algorithm* for Paging must decide which page to evict without knowledge of future requests, while an *off-line algorithm* can decide based on the whole sequence. On-line algorithms for Paging have been studied from a *competitive analysis* point of view in [16], comparing their performance to that of the optimal off-line algorithm. In that work it was shown that, if the cache can hold k pages, no deterministic on-line algorithm can be better than k-competitive, that is, guarantee less than k times the optimal off-line number of page faults on every input; it was also shown that some previously known

[1] This research was supported in part by EC project DYNDATA under program KIT, and by UBACyT projects "Algoritmos Eficientes para Problemas On-line con Aplicaciones" and "Modelos y Técnicas para Problemas de Optimización Combinatoria." Part of the material presented in this paper appeared in [8] and [9].
[2] Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, (1428) Capital Federal, Argentina. {efeuerst,asdel}@dc.uba.ar.

Received May 5, 1997; revised June 22, 1998. Communicated by A. Borodin.
Online publication October 12, 2001.

on-line algorithms such as Least-Recently-Used (LRU) and First-In-First-Out (FIFO) achieve that bound.

In this paper we introduce the *Multi-Threaded Paging* problem (MTP). MTP generalizes Paging to the case in which there is not just one sequence of requests but possibly many threads. This models situations in which the requests come from more than one independent source. Hence, apart from deciding *how* to serve a request, at each stage it is necessary to decide *which* request to serve among several possibilities. In this case there is no notion of "sequence of requests" but a more complex pattern that is not captured by the most general classes of on-line problems proposed in the literature (such as Metrical Task Systems [5] or Request–Answer Games [3], [12]).

MTP may be interesting from a practical point of view, as it models the situation arising in multi-tasking systems, where w independent processes simultaneously present their requirements of pages of secondary memory that must be brought into fast memory. At each moment, the system can see only one request per process, precisely the first unserved request of the sequence of requests that the process presents. Only after serving the current request of a particular process will the following request of that process be presented. The system must decide at each step whose request to satisfy, and also (as in normal Paging) which page of fast memory to remove on a page fault. In other words, it acts as scheduler and fast memory manager at the same time. The total number of page faults depends therefore not only on the strategy used to determine how each request is served but on which requests are satisfied, and when (in which order) this is done.

The problem that we introduce is also theoretically significant since it is, as far as we know, the first on-line problem in the literature in which there are several servers (each slot in the cache) and several clients (one for each thread). Indeed, this is the first paper that considers from a competitive analysis point of view some new issues arising in the family of multi-threaded problems, such as fairness restrictions and infinite inputs. In addition, all our definitions may be easily extended to infinite multi-threaded versions of, for example, Metrical Task Systems. This is the first step toward establishing an adequate model for analyzing the performance of on-line algorithms for many other interesting on-line problems with finite or infinite multi-threaded inputs.

The basic definition of MTP suggests two "dimensions" in which the problem can be analyzed. The first one regards *fairness*. Algorithms may simply try to minimize the number of page faults done while serving a set of w sequences of requests, or we may impose additional fairness restrictions, so all algorithms must guarantee that the next request of each thread will be served within a predetermined finite time.

The second dimension is related to *finiteness*. We can consider two different classes of inputs, namely finite and infinite input sequences. We show that the two definitions lead to subtly different results. This is in contrast to what happens in the single-threaded case, for which a paper by Raghavan and Snir [14] compared two alternative definitions of competitiveness (with finite and infinite input sequences respectively) and showed that both approaches are equivalent in a deterministic setting.

We consider three definitions of competitiveness for infinite inputs. They can be summarized as follows: The first one assumes that an algorithm that serves infinite inputs must behave efficiently at every finite stage of the process. The second and third definitions aim at comparing the algorithms' steady-state performances rather than to pick a single moment and count page faults up to that moment. In other words, algorithms

that treat infinite inputs are not compared at finite stages but when the number of served requests tends to infinity. The difference between the last two models is given by the way in which infinite inputs with bounded cost are treated. A priori, the three models of competitiveness seem different. We study them and find, however, that they are essentially equivalent. This suggests that the competitiveness results that we obtain reflect the intrinsic difficulty of the problem and are not a consequence of a too strict definition of competitiveness.

Four different on-line problems arise depending on whether we consider fairness restrictions or not, with finite or infinite input sequences. We study all of them, proving lower and upper bounds for the competitiveness of on-line algorithms. The main competitiveness results presented in this paper state that when no fairness restrictions are imposed it is possible to obtain competitive algorithms for finite and infinite inputs. On the other hand, for the fair problems in general there exist no competitive algorithms (although there is an interesting exception in the particular case of extremely tight fairness restrictions).

Fiat and Karlin [10] have considered a problem related to MTP, in which the input corresponds to a multi-pointer walk on an *access graph* [4]. Within that framework, the multiple threads of requests are merged in one input sequence, corresponding to an interleaved execution of the different threads. The way in which the sequences are interleaved in [10] is decided in an earlier stage of the process (and is the same for all algorithms). In contrast, in MTP each algorithm is free to decide (up to a certain limit, in the case of fairness restrictions) how to interleave the sequences.

Recently, Alborzi et al. [1] have proposed a multi-threaded version of the 1-server problem. Although the 1-server problem is a generalization of Paging with $k = 1$, only finite input sequences are considered in [1], and fairness restrictions are not explicitly modeled.

Kimbrel [13] analyzed the problem of deciding how to interleave independent sequences of operations of two kinds: input/output data prefetches and the consumption of the data by a processor. In this problem the goal is to minimize the processor's stall time on data fetches. The model is based on the fact that in many applications some lookahead can be assumed, i.e., input/output demands are disclosed to the file system in advance.

Further work related to the framework that we present in this paper has been done by Strejilevich de Loma [17], who considered some interesting particular cases; by Feuerstein et al. [6], who improved some of the results for the finite version; and by Seiden [15], who gave randomized lower and upper bounds for the problem under the definitions of competitiveness given in [8].

The remainder of this paper is organized as follows: In Section 2 we formally introduce the four different versions of our problem, that is, the finite and infinite versions, with and without fairness restrictions. Moreover, we introduce the different definitions of competitiveness that may be used for evaluating the performance of on-line algorithms for the different versions of MTP. Section 3 presents some basic material related to normal Paging that is used later. Sections 4 and 5 are devoted to the study of the finite and infinite versions of MTP, respectively. There we establish lower and upper bounds with and without fairness restrictions, under the different competitiveness models. Finally, Section 6 is dedicated to describing conclusions, further research, and open problems.

2. Different Problems and Performance Measures. In this section we present the different on-line problems that arise depending on whether we consider finite or infinite input sequences and whether we impose fairness restrictions or not. We also present three alternative definitions of competitiveness that can be applied to infinite multi-threaded problems.

2.1. *Finite versus Infinite Sequences*. In our first problem, called Finite-MTP (FMTP), algorithms are faced with a certain number of *finite* sequences of requests that have to be served *completely*, that is, algorithms have to arrive at the end of each one of the sequences. FMTP is given by the set of pages U and two positive integers k and w, the size of the cache and the number of sequences, respectively. $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_w$ is an *input tuple*, where each σ_i is a finite sequence of requests (each request is an element of U). The tuple formed by the jth request of each sequence is called the jth *row* of requests. At any stage, a sequence σ_i whose last request has not been served is called *active*.

At every moment, an algorithm for FMTP holds k distinct pages in its cache, and to serve a request to page $p \in U$, p must be present in the cache. An algorithm for FMTP receives a tuple of sequences σ as input and produces as output an interleaving of the sequences of σ together with a sequence of configurations of the cache used to serve each of the requests. We call this output a *schedule* for σ. The cost of a schedule is the sum of the Hamming distances between successive cache configurations of the schedule. An on-line algorithm must produce the schedule with the restriction that each configuration must be determined only as a function of the currently seen requests and all the requests already served by the algorithm. An off-line algorithm can decide each configuration based on the entire input.

Certain types of systems are formed by a finite number of infinite sequential processes that run concurrently, each one of them producing its own sequence of requests to memory pages. We can model these situations by considering infinite sequences of requests, and we get the infinite version of the problem, Infinite-MTP (IMTP). The task of an algorithm for this problem is the same as for the finite case, except from the fact that an infinite schedule must be produced. In this case the cost of a schedule is an infinite sum.

In both cases the goal of an algorithm is to produce a schedule that minimizes the total cost. In Section 2.3 we discuss how competitive analysis can be used to evaluate the performance of on-line algorithms for these problems.

2.2. *Fairness*. In Section 2.1 we did not make any consideration regarding the fairness of the algorithms for MTP. In other words, algorithms for both variants of the problem would try to minimize the number of page faults. For example, in the infinite case that could be achieved even by serving only the requests coming from one particular thread. In certain frameworks, such as multi-tasking systems, this would not be admissible. Therefore, it is natural to impose fairness restrictions. We model them by considering, as part of the input of the problem, a (usually large) integer t such that no request can "wait" more than t units of time from the moment the previous request of the sequence has been satisfied. Time is measured in the following way: one unit of time elapses each time the system serves a request. Consequently, we define the notion of a *t-fair* schedule.

An algorithm for the Fair-MTP problem must produce a t-fair schedule for the input tuple of sequences. An on-line algorithm must obviously produce the schedule based

only on the known part of the input. We can apply this notion to both finite and infinite inputs, and therefore we get two new problems, namely Fair-FMTP and Fair-IMTP.

Turning back to the definition, note that the problem is well defined whenever $t \geq w - 1$. An interesting particular case arises when $t = w - 1$. This corresponds to imposing that, after choosing an ordering among the sequences, the input must be served in a round-robin fashion.

For notational convenience, from now on we refer to the two problems defined in Section 2.1 as Unfair-FMTP and Unfair-IMTP, while the terms FMTP and IMTP are used when fairness is not relevant. Analogously, we use the terms Unfair-MTP and Fair-MTP (without reference to the finiteness of the sequences) when finiteness is irrelevant.

2.3. *Performance Measures*. For the finite versions of MTP, every algorithm must serve all the requests that appear in the input tuple. Therefore, the straightforward extension of the traditional definition of competitiveness naturally applies to this case. We say that an algorithm A for FMTP is c-competitive if and only if there exists a constant D such that for any input tuple σ, we have $C_{\mathrm{A}}(\sigma) - c \cdot C_{\mathrm{OPT}}(\sigma) \leq D$, where $C_{\mathrm{A}}(\sigma)$ is the cost incurred by A, and $C_{\mathrm{OPT}}(\sigma)$ is the cost incurred by an optimal off-line algorithm.

When we deal with infinite input sequences, things are less immediate: as the cost of a schedule is an infinite sum, it can be measured at finite moments or in the limit, and there may be different ways of comparing performances of algorithms. The previous definition of competitiveness may be naturally adapted to IMTP as follows.

MODEL *every-step*. An algorithm for IMTP is c-competitive if and only if

$$(\exists D)(\forall \sigma)(\forall \ell) C_{\mathrm{A}}(\sigma, \ell) - c \cdot C_{\mathrm{OPT}}(\sigma, \ell) \leq D,$$

where $C_{\mathrm{A}}(\sigma, \ell)$ is the cost incurred by the algorithm for serving a finite number ℓ of requests, and $C_{\mathrm{OPT}}(\sigma, \ell)$ is the cost incurred by an optimal off-line algorithm that serves ℓ requests.

On-line algorithms are not aware of the moment when their performance will be compared with that of the optimal off-line algorithm. Therefore this definition implies that an on-line algorithm, to be competitive on infinite sequences, must be competitive at every finite stage. This contrasts with the intuition that, for infinite inputs, one would like to evaluate an algorithm's performance in the long run, and not at particular moments. This point is addressed by the following definition.

MODEL *constant-limit*. An algorithm for IMTP is c-competitive if and only if

$$(\exists D)(\forall \sigma) \limsup_{\ell \to \infty} C_{\mathrm{A}}(\sigma, \ell) - c \cdot C_{\mathrm{OPT}}(\sigma, \ell) \leq D.$$

This second definition seems somehow too strict, as it requires the existence of one particular constant to bound the difference between the on-line cost and c times the optimal off-line cost. For costs that tend to infinity (something that may happen in IMTP) it could suffice that that difference is bounded. That is the intention of the third (and last) definition of competitiveness we present.

MODEL *variable-limit.* An algorithm for IMTP is c-competitive if and only if

$$(\forall \sigma) \limsup_{\ell \to \infty} C_{\mathrm{A}}(\sigma, \ell) - c \cdot C_{\mathrm{OPT}}(\sigma, \ell) < \infty.$$

An immediate consequence of this definition is that for infinite inputs that can be served with constant cost, any algorithm that eventually does not fault anymore will fit the definition. This is not true for the other two models.

In a classical paper [14], Raghavan and Snir compared two alternative definitions of competitiveness for Paging algorithms. The first one measures the performance of algorithms "in the limit," i.e., when the number of requests to serve tends to infinity, while the second one does so for finite sequences. By definition of the Paging problem, at every finite moment every algorithm has served the same set of requests, and that is the intuitive reason that makes both approaches essentially equivalent in a deterministic setting, as is proved in [14]. With multiple threads that reason is not valid anymore. Nevertheless, we will prove that the three models of competitiveness, that a priori seem different, are essentially equivalent. Notice that for $w = 1$ (normal Paging) our every-step and variable-limit models correspond respectively to the definitions of competitiveness on finite sequences and competitiveness in the limit, defined in [14]. In addition, for $w = 1$ there is no distinction between FMTP and the every-step model of IMTP.

3. Basics on Normal Paging.

In this section we present some basic material related to normal Paging, that we use in the following sections.

3.1. *The Paging Algorithm Flush-When-Full.*

All the algorithms presented in this paper for the different versions of MTP are based on Flush-When-Full (FWF), a very well known k-competitive on-line algorithm for Paging, that was introduced in [11]. However, it is worthwhile noting that we could have used any deterministic marking algorithm (for instance, LRU); we have chosen FWF to simplify the analysis.

FWF maintains a set of marked pages. Initially no page is marked. On each request, an unmarked page is evicted to make room for the requested page if necessary; in any case the requested page is marked. The behavior of FWF on a request to page p is shown in Figure 1. FWF works in *phases*, the first phase starting with the first request of the sequence and each new phase starting with the request that would have caused more than k pages to be marked (when the marks are deleted). It is easy to verify that FWF

```
If p is not present in the cache then
   If k pages are marked then
      % a new phase starts
      Erase all the marks
   end If
   Choose a page p' deterministically among the unmarked pages of the cache
   Evict p' and bring p
end If
Mark p.
```

Fig. 1. Behavior of algorithm Flush-When-Full on a request to page p.

has a cost of at most k per phase. On the other hand, in any phase k distinct pages are requested, and the first requested page of the next phase is different from those k pages; this implies that any algorithm must have at least one fault per phase.

3.2. *Off-Line Service*

LEMMA 1. *Let σ be a sequence of $d-1$ requests, chosen from a set of at most d distinct pages, with $d > k$. There exists an off-line algorithm that, starting with k of those pages in the cache, serves σ with at most $d-k$ page faults.*

PROOF. The algorithm is Farthest-in-the-Future (FF), also called MIN because it is an optimal off-line algorithm for Paging [2]. Each time a page fault occurs, FF evicts the page that appears farthest in the unserved part of the input sequence.

Suppose that every page that FF evicts is never requested again. This means that every request that produces a fault corresponds to a page that FF never had in the cache before the moment in which the fault occurs. Since there are at most $d-k$ pages that initially FF does not have in the cache, the cost of FF cannot exceed $d-k$.

On the other hand, consider the last request r that produces the eviction of a page p that is requested later. From the behavior of FF we know that p is evicted because the other $k-1$ pages of the cache appear before p in the unserved part of σ. Since r is the last request that produces the eviction of a page that is requested later, those $k-1$ pages will be in the cache of FF when requested, and then FF does not fault at least $k-1$ times. Thus in this case the cost of FF is at most $|\sigma| - (k-1) = d-k$. □

COROLLARY 2. *Let σ be a sequence of requests, chosen from a set of at most d distinct pages, with $|\sigma| \geq k$ and $d > k$. There exists an off-line algorithm that serves σ with at most $k + (d-k)\lceil (|\sigma|-k)/(d-1)\rceil$ page faults.*

PROOF. Consider an off-line algorithm that starts by loading into the cache the first k different pages that appear in σ. With this content in the cache, the algorithm can serve at least the first k requests of σ, with a total cost of k up to that moment. After that, the algorithm splits the (at most $|\sigma|-k$) remaining requests into sectors of $d-1$ consecutive requests each, and serves each sector using FF. By Lemma 1, the new algorithm has at most $d-k$ page faults in each sector, which proves the claim. □

4. Finite Multi-Threaded Paging.

In this section we consider FMTP, the finite version of MTP. In this version of the problem each sequence contains a finite number of requests, and every algorithm must serve all those requests. We start with a technique useful for obtaining lower bounds on the competitiveness of on-line algorithms for both the fair and unfair cases of FMTP. After that, we analyze them separately.

4.1. *How to Prove Lower Bounds.*

The following lemma is a rephrasing of a technique that has been used often in the literature of single-threaded on-line problems.

LEMMA 3. *Let A be any on-line algorithm for FMTP. If there exist a constant E and an infinite family $\mathcal{F}$ of input tuples σ such that*

1. *for all $\sigma \in \mathcal{F}$ we have $C_{\mathrm{A}}(\sigma) \geq cC_{\mathrm{OPT}}(\sigma) + E$, and*
2. *the value of $C_{\mathrm{A}}(\sigma)$ is not upper bounded in $\mathcal{F}$,*

then A is not better than c-competitive.

PROOF. The proof follows by contradiction. Suppose that the conditions hold and that A is $(c - \varepsilon)$-competitive for some $\varepsilon > 0$. Then there exists a constant D such that, for all $\sigma \in \mathcal{F}$,

$$D \geq C_{\mathrm{A}}(\sigma) - (c - \varepsilon)C_{\mathrm{OPT}}(\sigma) \geq cC_{\mathrm{OPT}}(\sigma) + E - (c - \varepsilon)C_{\mathrm{OPT}}(\sigma) = E + \varepsilon C_{\mathrm{OPT}}(\sigma),$$

that is, $C_{\mathrm{OPT}}(\sigma)$ is upper bounded in $\mathcal{F}$. However, by hypothesis $C_{\mathrm{A}}(\sigma)$ is unbounded in $\mathcal{F}$, so A cannot be competitive, a contradiction. □

4.2. *The Unfair Finite Problem.* Although in FMTP every algorithm must serve the same set of requests, the order in which those requests are served is decided by each algorithm. We use this property to obtain a lower bound for Unfair-FMTP. The lower bound does not match the upper bound we present later; however, if we restrict to normal Paging ($w = 1$) it is the best possible (k). As is usually done in competitive analysis, we compare on-line strategies with an *adversary* that chooses the input and serves it optimally.

THEOREM 4. *No on-line algorithm for Unfair-FMTP is better than $(k + 1 - 1/w)$-competitive.*

PROOF. We use Lemma 3 to prove the lower bound. Let A be any on-line algorithm. Let $m = nk$, where n is any positive integer. Consider a set U of $w(k + 1)$ pages, partitioned into w disjoint subsets $U_1, U_2, \ldots, U_w$, each one of $k + 1$ pages. Since k is the size of the cache, in any configuration of A there is at least one page of each U_i missing from A. The first request of each sequence σ_i is to a page of U_i that is not present in the initial configuration of A. Each new request of σ_i is to a page of U_i not in A's cache after the algorithm serves the previous request of the same sequence. Let σ_j be the last sequence of which A serves the mth request. The other sequences have exactly m requests, while σ_j continues with the concatenation in any order of the other sequences, e.g., $\sigma_1, \sigma_2, \ldots, \sigma_{j-1}, \sigma_{j+1}, \ldots, \sigma_w$ (see Figure 2). By construction A faults on each one of the first m requests of each sequence, with a cost of at least wm. When A serves the mth request of σ_j, it must still serve the remaining requests of that sequence. This cannot require less than the cost incurred by an optimal off-line algorithm that serves those requests starting with the same cache as A. If we denote as R that optimal cost, we have

$$C_{\mathrm{A}}(\sigma) \geq wm + R.$$

Let C^i_{OPT} be the optimal off-line cost for serving the first m requests of σ_i. By Corollary 2 we know that $(\forall i)\ C^i_{\mathrm{OPT}} \leq s = k + n - 1$, and then the optimal off-line cost for serving

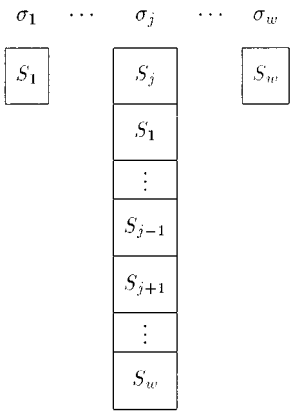

Fig. 2. Sequences used in the proof of Theorem 4.

all the requests is

$$C_{\mathrm{OPT}}(\sigma) \leq \sum_{i=1}^{w} C^i_{\mathrm{OPT}} \leq ws,$$

because the adversary can serve σ_j and at the same time serve the corresponding requests of the other sequences (with no additional cost for those last sequences). It is also true that

$$C_{\mathrm{OPT}}(\sigma) \leq C^j_{\mathrm{OPT}} + k + R \leq s + k + R,$$

because the adversary can serve the first m requests of σ_j, and then serve the remainder of that sequence starting with the same cache as A. Since $wm = wks - wk(k-1)$, using standard calculations we obtain

$$\begin{aligned} C_{\mathrm{A}}(\sigma) &\geq wm + R \geq wm + C_{\mathrm{OPT}}(\sigma) - s - k \\ &= (k - 1/w)ws + C_{\mathrm{OPT}}(\sigma) - wk(k-1) - k \\ &\geq (k - 1/w)C_{\mathrm{OPT}}(\sigma) + C_{\mathrm{OPT}}(\sigma) - wk(k-1) - k \\ &= (k + 1 - 1/w)C_{\mathrm{OPT}}(\sigma) - wk(k-1) - k. \end{aligned}$$

This and the fact that $C_{\mathrm{A}}(\sigma)$ is unbounded for the family of instances, imply the result by Lemma 3. □

We now present an on-line algorithm for Unfair-MTP. We call the algorithm Alternating–Flush-When-Full (AFWF), and it is described in Figure 3. The algorithm works in rounds; each round consists of applying a phase of FWF to each sequence $\sigma_1, \sigma_2, \ldots, \sigma_w$. In Unfair-FMTP the algorithm must check whether the sequences are over. We will see now that AFWF is wk-competitive for that problem.

THEOREM 5. *Algorithm AFWF is wk-competitive for Unfair-FMTP.*

```
While there is at least one request to be served do
      % a new round starts
      i ← 1
      While i ≤ w do
            % a new phase starts
            Apply FWF to σ_i for a phase or till the sequence is over
            i ← i + 1
      end While
end While.
```

Fig. 3. Algorithm Alternating–Flush-When-Full.

PROOF. By definition of AFWF, the cost of each one of the phases of a round is at most k, and thus the cost of each round is at most wk. Suppose that for serving all the requests AFWF stopped in the $(m+1)$st round, eventually with the last round incomplete. Then its total cost is

$$C_{\mathrm{AFWF}}(\sigma) \leq wkm + wk.$$

Let σ_j be a sequence that finished when AFWF was in its last round. Restricted to σ_j, in any round k distinct pages are requested, and the first requested page of the next round is different from those k pages. Hence the adversary must have at least one fault per completed round, and then we have

$$C_{\mathrm{OPT}}(\sigma) \geq m,$$

which implies

$$C_{\mathrm{AFWF}}(\sigma) \;\leq\; wkC_{\mathrm{OPT}}(\sigma) + wk. \qquad \square$$

Let A be any on-line algorithm for Paging. If we use A to serve completely each one of the w sequences of requests, we obtain an on-line algorithm for Unfair-FMTP. It is not difficult to show that if A is c-competitive, then the resulting algorithm for Unfair-FMTP is wc-competitive. This means that any k-competitive Paging algorithm defines a wk-competitive algorithm for Unfair-FMTP. However, we prefer AFWF because algorithms of the type just described can behave very "unfairly"; in addition, AFWF can be used in the infinite problem.

Recently, Feuerstein et al. [6] have obtained a stronger $\Omega(k \log w)$ lower bound for Unfair-FMTP. For completeness we present the result, without a proof.

THEOREM 6. *Let $c < (0.5\lfloor \log w \rfloor + 1)k/e$. Let D be any constant. For every on-line algorithm A for Unfair-FMTP there exists an input tuple σ such that*

1. $|\sigma| - cC_{\mathrm{OPT}}(\sigma) > D$, *and*
2. *for every $\ell \leq |\sigma|$ the cost of A for serving ℓ requests from σ is at least ℓ (in other words, σ is the nemesis for A).*

COROLLARY 7. *No on-line algorithm for Unfair-FMTP is better than $[(0.5\lfloor \log w \rfloor + 1)k/e]$-competitive.*

4.3. *The Fair Finite Problem.* We now analyze FMTP in the case in which fairness restrictions are explicitly imposed. Recall that fairness is modeled by considering an integer t such that at most t other requests can be satisfied since the moment any request is seen till the moment that request is served. The minimum possible value for t is $w-1$.

A straightforward lower bound for the competitiveness of on-line algorithms for Fair-FMTP can be obtained by considering an instance formed by w identical sequences, each one like the sequence used in the proof of the lower bound for normal Paging [16]. The following theorem formalizes this argument.

THEOREM 8. *No on-line algorithm for Fair-FMTP is better than k-competitive, even if we restrict the sequences of requests to be formed by at most $k+1$ distinct pages.*

PROOF. Let A be any on-line algorithm. Let U be a set of $k+1$ distinct pages. We construct the input tuple by rows, all the requests in each row to the same page. The requests of the first row are to a page of U not in A's cache at the beginning, and the requests of each new row are to a page of U not in A's cache after the algorithm serves the first request of the previous row. This guarantees that A faults at least once per row. The adversary chooses any fixed order for the sequences, and cyclically serves one request of each sequence. This behavior is fair for any $t \geq w-1$, and allows the adversary to fault only once every k rows. □

Without fairness restrictions, a smart on-line algorithm for FMTP could delay serving sequences that look "too hard." When fair behavior is imposed, an on-line algorithm can be forced to serve with high cost requests that could be served more efficiently by delaying them. The following theorem exploits this fact to prove that in general there is no competitive on-line algorithm for Fair-FMTP.

THEOREM 9. *There is no competitive on-line algorithm for Fair-FMTP with $t \geq w \geq 2$, even if we restrict the sequences of requests to be formed by at most $k+1$ distinct pages.*

PROOF. Let A be any on-line algorithm. Given any pair of constants c and D, we will show an input tuple σ for which $C_{\mathrm{A}}(\sigma) - cC_{\mathrm{OPT}}(\sigma) > D$. This will prove the claim. Let n be a "big" positive integer that we fix later. Let $U = \{a, b_1, b_2, \ldots, b_k\}$ be a set of $k+1$ distinct pages. Let $U' = U - \{a\} = \{b_1, b_2, \ldots, b_k\}$. The first n requests of every sequence are to page a. Let σ_p and σ_q be two sequences such that A serves the nth request of σ_p before the algorithm serves the nth request of σ_q (remember that $w \geq 2$).

We choose the following requests so as to make σ very expensive if σ_p is served together with the initial part of σ_q, but cheap if several requests of σ_q are served first. The algorithm A will act the first way to satisfy the fairness constraint, and so it will have a high cost. The off-line adversary will delay the requests of σ_p until it has served the initial requests of σ_q, bounding its cost by a constant.

After the nth request of σ_p, the sequence continues with requests to pages $b_1, b_2, \ldots, b_k$ repeatedly. On the other hand, σ_q continues with n more requests to page a. The requests not mentioned above are to arbitrary pages in U' (see Figure 4).

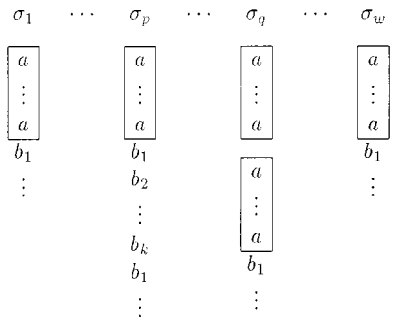

Fig. 4. Sequences used in the proof of Theorem 9.

Notice that, in every group of $t+1$ consecutive requests that any fair algorithm serves, it must satisfy at least one request and at most $t+1-(w-1)\times 1 = t-w+2$ requests of each sequence (assuming that all the w sequences are active). In every group of $(t+1)k$ consecutive requests, the number of served requests of each sequence is between k and $(t-w+2)k$. When A serves the nth request of σ_q, there are n pending requests to page a in that sequence, while σ_p repeats requests to pages $b_1, b_2, \ldots, b_k$. Before A terminates with all the requests to page a in σ_q, in every group of $(t+1)k$ consecutive requests the algorithm must satisfy at least k requests to page a in σ_q, and at least one request to each page $b_1, b_2, \ldots, b_k$ in σ_p. These are $k+1$ distinct pages, and hence the cost of A for serving each group of $(t+1)k$ consecutive requests is at least 1. If the sequences are long enough, this occurs a minimum of $\lfloor \frac{n}{(t-w+2)k} \rfloor$ times until A serves all the requests to page a in σ_q. Therefore the total cost of A for sufficiently large sequences is

$$C_{\mathrm{A}}(\sigma) \geq \left\lfloor \frac{n}{(t-w+2)k} \right\rfloor.$$

We now consider the adversary's cost. Being that $t \geq w$, for each $w+1$ requests the adversary can serve one request of σ_q and one request of each one of the w sequences (including σ_q). This behavior ensures that in the first $(w+1)n$ steps the adversary serves the $2n$ requests to page a in σ_q, and the n requests to that page in the other sequences. The cost for doing that is at most 1. All the other requests are to pages in U'. The size of the cache is k and there are k pages in U'. Then the adversary can serve the remaining requests with a cost of at most k, and therefore we have

$$C_{\mathrm{OPT}}(\sigma) \leq 1+k.$$

To conclude the proof we only need to fix the value of n. Define $n(x,y) = (t-w+2)k\lceil x + xk + y\rceil$, and choose any $n \geq n(c, D+1)$. Then we have

$$\begin{aligned} C_{\mathrm{A}}(\sigma) - cC_{\mathrm{OPT}}(\sigma) &\geq \left\lfloor \frac{n}{(t-w+2)k} \right\rfloor - c(1+k) \\ &\geq \lceil c + ck + D + 1\rceil - c(1+k) \geq D+1 > D, \end{aligned}$$

and the result follows. □

Extremely tight fairness restrictions. The preceding theorem does not cover the cases of Fair-FMTP in which $w = 1$ or $t = w - 1$. With $w = 1$ fairness restrictions have no sense, and we are faced with regular Paging. On the other hand, the case $t = w - 1$ requires further analysis. It is worthwhile distinguishing two different situations of Fair-FMTP with $t = w - 1$. The first one is while all the sequences are active, and the other one is when one or more sequences have finished.

While all the sequences are active the algorithms (on-line or not) must apply a round-robin strategy, i.e., serve one request of each sequence in a fixed order which is repeated over and over again. This implies that after each w new requests, the set of requests served by any algorithm is the same. This particular situation of Fair-FMTP is closely related to normal Paging, since after an algorithm has chosen the order in which to serve the requests, we can think that there is only one sequence to be served, as is the case in Paging. Nevertheless, the following points show that the two problems are different:

- In Fair-FMTP we can say that any algorithm has served the same set of requests only after each w new requests (after each row of requests), not at every step as in Paging.
- Algorithms (of any type) can choose between $w!$ distinct orders of the sequences. However, off-line algorithms can choose with information on the whole input tuple, while on-line algorithms must base their decisions only on the first requests of each sequence. Moreover, any choice made by an on-line algorithm can be fooled if the first two requests of all the sequences are to the same page. In other words, from a competitive analysis point of view, off-line algorithms can really decide in which order to serve the sequences, while the choice of an on-line algorithm is an illusion.
- On-line algorithms can see the following w requests to serve, not only one of them. More precisely, on-line algorithms can make use of a lookahead of size w. This is not a clear advantage for on-line algorithms, since this kind of lookahead can be easily neutralized by replacing each request with w requests to the same page.

As soon as a sequence finishes in Fair-FMTP with $t = w - 1$, distinct algorithms can serve the remaining requests in very different ways. This can be used to extend Theorem 9 to the case $t = w - 1$ and $w \geq 3$. The idea of the proof consists of making the on-line algorithm serve a first sequence of length 1.

COROLLARY 10. *There is no competitive on-line algorithm for Fair-FMTP with* $t = w - 1$ *and* $w \geq 3$ ($t = w - 1 \geq 2$), *even if we restrict the sequences of requests to be formed by at most* $k + 1$ *distinct pages*.

PROOF. Let A be any on-line algorithm. Let $U = \{a, b_1, b_2, \ldots, b_k\}$ be a set of $k + 1$ distinct pages. The first request of every sequence is to page a. Since $t = w - 1$, any algorithm must start by completely serving the first row of requests. The first sequence that A serves terminates immediately. Thus, at the end of the first row of requests, the number of active sequences is $w' = w - 1$, with $t = w' \geq 2$. From that point on, we repeat the construction of Theorem 9. Reasoning as in that theorem we obtain that the total cost of A is at least $\lfloor \frac{n}{(t-w'+2)k} \rfloor$.

```
While there is at least one request to be served do
      % a new super-phase starts
      Apply a phase of FWF to the sequence σ* formed by taking in turn
      one request of each active sequence σ_1, σ_2, ..., σ_w
      Serve the next request of σ* (no matter how)
      Serve all the pending requests of σ* in the same row
      of the last served request (no matter how)
end While.
```

Fig. 5. Algorithm Round-Robin–Flush-When-Full.

The adversary can serve the first row of requests in the order that A does. After that, for each $w' + 1$ requests the adversary can serve one request of σ_q and one request of each one of the w' active sequences (including σ_q). With this behavior the adversary honors the fairness constraint, and has a total cost of at most $1 + k$.

The proof concludes as in Theorem 9, by choosing n as large as needed. □

We have seen in Theorem 9 and Corollary 10 that the only case of Fair-FMTP in which competitive on-line algorithms can exist is when $t = w - 1$ and $w = 2$. For this case the lower bound of Theorem 8 holds; a competitive on-line algorithm is presented now.

A possible on-line algorithm for Fair-MTP is Round-Robin–Flush-When-Full (RRFWF), which is described in Figure 5. The algorithm works in "super-phases"; each super-phase consists of applying a phase of FWF to the sequence formed by taking in turn one request of each sequence $\sigma_1, \sigma_2, \ldots, \sigma_w$, and then serving the next request and all the other pending requests in the same row; these additional requests are served by RRFWF in an arbitrary deterministic way. In the finite problem the algorithm must check for end-of-sequence. Note that due to the additional requests served in each super-phase, when $w = 1$ RRFWF *is not* equivalent to FWF. Clearly, RRFWF is fair for any legal value of t. We show now that it is $(k + w)$-competitive for Fair-FMTP with $t = w - 1$ and $w = 2$.

THEOREM 11. *Algorithm RRFWF is $(k+w)$-competitive for Fair-FMTP with $t = w-1$ and $w = 2$.*

PROOF. By definition of RRFWF, the cost of each super-phase does not exceed $k + 1 + (w - 1) = k + w$. Suppose that for serving all the requests RRFWF stopped in the $(m + 1)$st super-phase, eventually with the last super-phase incomplete. Hence we have for its total cost

$$C_{\mathrm{RRFWF}}(\sigma) \leq (k + w)m + k + w.$$

Note that any algorithm must serve the requests row by row: while all the sequences are active, this is true due to the fact that $t = w - 1$; when one of the (two) sequences has finished, this trivially holds. Therefore, at the end of each one of the completed super-phases the adversary must have served the same requests as RRFWF, because RRFWF only terminates a super-phase when it has completely served some row. Since in each

super-phase at least $k+1$ distinct pages appear, the adversary must fault at least once in each completed super-phase, and then we have

$$C_{\mathrm{OPT}}(\sigma) \geq m,$$

so

$$C_{\mathrm{RRFWF}}(\sigma) \leq (k+w)C_{\mathrm{OPT}}(\sigma) + k + w. \qquad \square$$

We must point out that no general on-line algorithm for Fair-FMTP beats RRFWF: in [17] it was proved that no on-line algorithm for Fair-FMTP with $t = w - 1$, even w, and $k = 1$ is better than $(k+w)$-competitive.

5. Infinite Multi-Threaded Paging. In this section we analyze IMTP, that is, MTP with infinite input sequences. As we did in Section 4, we first present some general results about the problem. After that we consider separately the fair and unfair cases. For each one of these cases we analyze IMTP under the three competitiveness models introduced in Section 2.

Some of the results included in this section establish distinct equivalencies between the competitiveness models. We mainly use two types of arguments to establish those results. The first one is a *replacement* argument: given an on-line algorithm and an input tuple in which the algorithm behaves poorly, we replace some of the unserved requests of the input tuple with "cheap" requests; the on-line algorithm still has a bad performance in the modified input tuple, even when the number of requests to serve tends to infinity. The second type is a *concatenation* argument: given an on-line algorithm, we concatenate parts of input tuples (in which the algorithm behaves poorly) to form another input tuple; this last input tuple is finally presented to the on-line algorithm. A concatenation argument was used in [14] to show that the competitiveness achievable by deterministic on-line algorithms for Paging is the same on finite sequences and in the limit. Nevertheless, the argument does not translate directly to MTP, since at each step different algorithms may have served different sets of requests.

5.1. *General Results.* It is easy to see that our three performance measures are monotonically weaker. We use this fact repeatedly within this section.

THEOREM 12. *Let A be any algorithm for IMTP. If A is c-competitive under the every-step model, then it is c-competitive under the constant-limit model.*

PROOF. The proof follows directly from the definitions of the two models. $\square$

THEOREM 13. *Let A be any algorithm for IMTP. If A is c-competitive under the constant-limit model, then it is c-competitive under the variable-limit model.*

PROOF. The proof follows directly from the definitions of the two models. $\square$

We now present tools that allow us to obtain lower bounds on the competitiveness of on-line algorithms for IMTP.

LEMMA 14. *Let A be any on-line algorithm for IMTP. If there exist a constant E and an infinite family $\mathcal{F}$ of pairs (σ, ℓ) such that*

1. *for all $(\sigma, \ell) \in \mathcal{F}$ we have $C_{\mathrm{A}}(\sigma, \ell) \geq cC_{\mathrm{OPT}}(\sigma, \ell) + E$, and*
2. *the value of $C_{\mathrm{A}}(\sigma, \ell)$ is not upper bounded in $\mathcal{F}$,*

then A is not better than c-competitive under the every-step model.

PROOF. Replace syntactically (σ) by (σ, ℓ) in the proof of Lemma 3. □

LEMMA 15. *Let A be any on-line algorithm for IMTP. If there exist a constant E and an infinite family $\mathcal{F}$ of pairs (σ, ℓ) such that*

1. *for all $(\sigma, \ell) \in \mathcal{F}$ and for all $\ell' \geq \ell$ we have $C_{\mathrm{A}}(\sigma, \ell') \geq cC_{\mathrm{OPT}}(\sigma, \ell') + E$, and*
2. *the value of $C_{\mathrm{A}}(\sigma, \ell)$ is not upper bounded in $\mathcal{F}$,*

then A is not better than c-competitive under the constant-limit model.

PROOF. As in Lemmas 3 and 14, the proof follows by contradiction. Consider a constant $\varepsilon > 0$. By hypothesis we know that for all $(\sigma, \ell) \in \mathcal{F}$ and for all $\ell' \geq \ell$,

$$\begin{aligned} C_{\mathrm{A}}(\sigma, \ell') - (c - \varepsilon)C_{\mathrm{OPT}}(\sigma, \ell') &\geq cC_{\mathrm{OPT}}(\sigma, \ell') + E - (c - \varepsilon)C_{\mathrm{OPT}}(\sigma, \ell') \\ &= E + \varepsilon C_{\mathrm{OPT}}(\sigma, \ell'). \end{aligned}$$

Now assume that A is $(c - \varepsilon)$-competitive. Then there exists a constant D such that

$$D \geq \limsup_{\ell' \to \infty} C_{\mathrm{A}}(\sigma, \ell') - (c - \varepsilon)C_{\mathrm{OPT}}(\sigma, \ell') \geq E + \varepsilon \limsup_{\ell' \to \infty} C_{\mathrm{OPT}}(\sigma, \ell').$$

Since the cost is a non-decreasing function of the number of served requests, the above expression implies that $C_{\mathrm{OPT}}(\sigma, \ell)$ is upper bounded in $\mathcal{F}$, preventing A from being competitive because $C_{\mathrm{A}}(\sigma, \ell)$ is unbounded in $\mathcal{F}$. □

5.2. *The Unfair Infinite Problem.* At each step of IMTP each algorithm may have served different requests. Based on this fact we now prove a lower bound for Unfair-IMTP under the every-step model.

THEOREM 16. *No on-line algorithm for Unfair-IMTP is better than wk-competitive under the every-step model.*

PROOF. We use Lemma 14 to prove the claim. Let A be any on-line algorithm. Let $m = nk$, where n is any positive integer, and consider a set U of pages, partitioned into w disjoint subsets $U_1, U_2, \ldots, U_w$, each one of $k + 1$ pages. The input tuple is constructed as in Theorem 4 until A serves $\ell = wm$ requests. In this way A necessarily faults on each one of these requests, and its cost is

$$C_{\mathrm{A}}(\sigma, \ell) \geq wm = wnk.$$

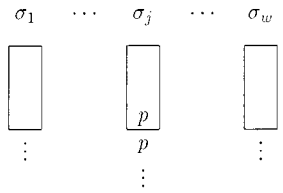

Fig. 6. Sequences used in the proof of Theorem 16.

Let σ_j be the last sequence of which A serves the mth request, or a sequence such that A does not serve its mth request at all. Now suppose σ_j continues with repeated requests to page p which is the mth request of σ_j (see Figure 6). By Corollary 2 the sequence σ_j can be served with at most $k + n - 1$ page faults, with no cost after the mth request. Since the adversary can just serve requests of σ_j forever, the optimal off-line cost is

$$C_{\mathrm{OPT}}(\sigma, \ell) \leq k + n - 1.$$

Then we have

$$C_{\mathrm{A}}(\sigma, \ell) \geq wnk \;=\; wk(k + n - 1) - wk(k - 1) \geq wkC_{\mathrm{OPT}}(\sigma, \ell) - wk(k - 1). \quad \square$$

We will see now that algorithm AFWF, proposed for Unfair-FMTP and described in Figure 3, is strongly competitive for Unfair-IMTP under the every-step model.

THEOREM 17. *Algorithm AFWF is wk-competitive for Unfair-IMTP under the every-step model.*

PROOF. The proof is very similar to that of Theorem 5. Suppose that after ℓ requests AFWF completed m rounds and is currently in the $(m + 1)$st round. Then its cost is

$$C_{\mathrm{AFWF}}(\sigma, \ell) \leq wkm + wk.$$

There must be at least one sequence for which the adversary served at least as many requests as AFWF in that sequence. Let σ_j be such a sequence. Restricted to σ_j, in any round k distinct pages are requested, and the first requested page of the next round is different from those k pages. This means that we can charge to the adversary at least a cost of 1 for every completed round, and hence we have

$$C_{\mathrm{OPT}}(\sigma, \ell) \geq m,$$

which implies

$$C_{\mathrm{AFWF}}(\sigma, \ell) \;\leq\; wkC_{\mathrm{OPT}}(\sigma, \ell) + wk. \quad \square$$

We have presented in Theorems 16 and 17 lower and upper bounds for Unfair-IMTP under the every-step model. We now extend those results to the constant-limit model.

THEOREM 18. *Let A be any on-line algorithm for Unfair-IMTP. Then the following statements are equivalent*:

(I) *A is c-competitive under the every-step model*;
(II) *A is c-competitive under the constant-limit model*.

PROOF. We proved in Theorem 12 that (I) $\Rightarrow$ (II). To prove that (II) $\Rightarrow$ (I), we will see that $\neg$(I) $\Rightarrow$ $\neg$(II). Let D be any constant. Since $\neg$(I) holds, there exist σ and ℓ such that

$$C_{\mathrm{A}}(\sigma, \ell) - cC_{\mathrm{OPT}}(\sigma, \ell) > D + 1 + c.$$

After serving ℓ requests, there must be at least one sequence σ_j for which the adversary has served at least the same number of requests as A in that sequence. The part of σ_j that none of the algorithms has served may contain requests only to some particular page. The adversary can serve any number of those requests with at most one new page fault. On the other hand, the cost of A for serving additional requests cannot decrease. Therefore, for any number of requests $\ell' \geq \ell$ we have

$$C_{\mathrm{A}}(\sigma, \ell') \geq C_{\mathrm{A}}(\sigma, \ell)$$

and

$$C_{\mathrm{OPT}}(\sigma, \ell') \leq C_{\mathrm{OPT}}(\sigma, \ell) + 1,$$

which implies

$$C_{\mathrm{A}}(\sigma, \ell') - cC_{\mathrm{OPT}}(\sigma, \ell') \geq C_{\mathrm{A}}(\sigma, \ell) - cC_{\mathrm{OPT}}(\sigma, \ell) - c > (D + 1 + c) - c = D + 1,$$

and then

$$\limsup_{\ell' \to \infty} C_{\mathrm{A}}(\sigma, \ell') - cC_{\mathrm{OPT}}(\sigma, \ell') \ \geq \ D + 1 > D. \qquad \square$$

So far we have shown that our on-line algorithm AFWF is wk-competitive for Unfair-IMTP under the every-step and constant-limit models, and we have provided a matching lower bound. Therefore Unfair-IMTP is closed under those two models. By Theorem 13 we know that AFWF is wk-competitive also under the variable-limit model. We were not able to prove whether AFWF is optimal (strongly competitive) under this last model. However, we will obtain from the finite version of the problem a lower bound for Unfair-IMTP under the variable-limit model.

LEMMA 19. *Given a constant c, if for every on-line algorithm A for Unfair-FMTP there exists a finite input tuple σ such that*

1. $|\sigma| - cC_{\mathrm{OPT}}(\sigma) \geq 1$, *and*
2. *for every $\ell \leq |\sigma|$ the cost of A for serving ℓ requests from σ is at least ℓ,*

then no on-line algorithm for Unfair-IMTP is c-competitive under the variable-limit model.

PROOF. Let IA be any on-line algorithm for Unfair-IMTP, and IADV an off-line adversary. Let ADV be an off-line adversary for Unfair-FMTP that is able to build an input tuple that fulfills the two conditions mentioned above. Let m be any positive integer. Roughly speaking we concatenate finite input tuples σ, so as to obtain an infinite input tuple σ_{IA} such that $C_{\mathrm{IA}}(\sigma_{\mathrm{IA}}, \ell) - cC_{\mathrm{OPT}}(\sigma_{\mathrm{IA}}, \ell)$ is not upper bounded.

The adversary IADV simulates ADV to decide the initial requests of each sequence of σ_{IA}. The first time that, following the construction of ADV, a sequence should terminate, IADV starts simulating a new instance of ADV to extend that sequence. This second ADV is also used to extend the other sequences when, according to the first ADV, they should terminate. In general, the first time that, following the construction of the ith ADV, a sequence should terminate, IADV starts simulating an $(i+1)$st ADV to extend the sequences. Each new ADV works over a completely new set of pages. Let $\sigma(i)$ be the (finite) part of σ_{IA} generated while simulating the ith ADV. By hypothesis, we can charge unitary cost to IA for each request it serves. Hence its cost for $s_m = \sum_{i=1}^{m} |\sigma(i)|$ requests is

$$C_{\mathrm{IA}}(\sigma_{\mathrm{IA}}, s_m) \geq s_m = \sum_{i=1}^{m} |\sigma(i)|.$$

The adversary IADV can optimally serve the requests of $\sigma(1)$, then the requests of $\sigma(2)$, and so on. Thus, it is clear that

$$C_{\mathrm{OPT}}(\sigma_{\mathrm{IA}}, s_m) \leq \sum_{i=1}^{m} C_{\mathrm{OPT}}(\sigma(i)).$$

Then we have

$$C_{\mathrm{IA}}(\sigma_{\mathrm{IA}}, s_m) - cC_{\mathrm{OPT}}(\sigma_{\mathrm{IA}}, s_m) \geq \sum_{i=1}^{m} [|\sigma(i)| - cC_{\mathrm{OPT}}(\sigma(i))] \geq \sum_{i=1}^{m} 1 = m.$$

Since m was any positive integer, the result follows. □

COROLLARY 20. *No on-line algorithm for Unfair-IMTP is better than* $[(0.5\lfloor \log w \rfloor + 1)k/e]$*-competitive under the variable-limit model.*

PROOF. The claim follows from Theorem 6 and Lemma 19. □

5.3. *The Fair Infinite Problem.* Theorem 9 states that in general there is no competitive on-line algorithm for Fair-FMTP. If we consider infinite input sequences in the proof of that theorem, it is straightforward to extend the result to Fair-IMTP under the every-step and constant-limit models. In fact, the result is valid under the three performance measures used for IMTP, as the following theorem shows.

THEOREM 21. *There is no competitive on-line algorithm for Fair-IMTP with* $t \geq w \geq 2$*, under the every-step, constant-limit, and variable-limit models, even if we restrict the sequences of requests to be formed by at most* $k+1$ *distinct pages.*

PROOF. By Theorems 12 and 13, it is enough to prove the claim for the variable-limit model. Let A be any on-line algorithm. Given any constant c, we will show an input tuple σ for which $C_{\mathrm{A}}(\sigma, \ell) - cC_{\mathrm{OPT}}(\sigma, \ell)$ is not upper bounded, proving the theorem. In order to build the input tuple we repeatedly apply the construction of Theorem 9. Let $n \geq n(c, 1) = (t - w + 2)k\lceil c + ck + 1\rceil$. Let $U = \{a, b_1, b_2, \ldots, b_k\}$ be a set of $k + 1$ distinct pages. Let $U' = U - \{a\}$. The first n requests of all the sequences are to page a. Let σ_{p_1} and σ_{q_1} be two sequences such that A serves the nth request of σ_{p_1} before the algorithm serves the nth request of σ_{q_1}. The sequence σ_{p_1} continues with requests to pages $b_1, b_2, \ldots, b_k$ repeatedly, while σ_{q_1} continues with n more requests to page a. Assume that the requests not mentioned above are to arbitrary pages in U'. Reasoning as in Theorem 9, it is not difficult to show the following facts: since the moment that A serves the nth request of σ_{q_1} until the moment that A satisfies all the requests to page a in that sequence, the algorithm has a cost of at least 1 in every group of $(t+1)k$ consecutive requests that it serves; moreover, this occurs a minimum of $\lfloor \frac{n}{(t-w+2)k} \rfloor$ times; on the other hand, the adversary can serve the complete input tuple with a cost of at most $1 + k$. Let ℓ_1 be the number of served requests when A satisfies the last request to page a in σ_{q_1}. Based on the above-mentioned facts, we have

$$C_{\mathrm{A}}(\sigma, \ell_1) - cC_{\mathrm{OPT}}(\sigma, \ell_1) \geq \left\lfloor \frac{n}{(t - w + 2)k} \right\rfloor - c(1+k) \geq \lceil c + ck + 1\rceil - c(1+k) \geq 1.$$

Now imagine (only for the purposes of the analysis) that A temporarily stops serving requests while the adversary continues until it has served more requests than A in all the sequences. The adversary can change the requests that it has not served, because A has not seen them. The modified part starts with n requests to page a in all the sequences. Now suppose that A continues serving requests, and let σ_{p_2} and σ_{q_2} be two sequences such that A serves the nth request of the modified part of σ_{p_2} before the algorithm serves the nth request of the modified part of σ_{q_2}. The sequence σ_{p_2} continues with requests to pages $b_1, b_2, \ldots, b_k$ repeatedly, while σ_{q_2} continues with n more requests to page a. Again the remaining requests are to arbitrary pages in U'. Let ℓ_2 be the total number of requests served by A when it satisfies all the requests to page a. Reasoning as in Theorem 9 for the modified part of the input tuple, we obtain

$$C_{\mathrm{A}}(\sigma, \ell_2) \geq C_{\mathrm{A}}(\sigma, \ell_1) + \left\lfloor \frac{n}{(t - w + 2)k} \right\rfloor$$

and

$$C_{\mathrm{OPT}}(\sigma, \ell_2) \leq C_{\mathrm{OPT}}(\sigma, \ell_1) + 1 + k,$$

and then

$$C_{\mathrm{A}}(\sigma, \ell_2) - cC_{\mathrm{OPT}}(\sigma, \ell_2) \geq 2.$$

The adversary can change the requests that A has not seen as many times as desired. This implies that for any positive integer m there is a number of requests ℓ_m such that

$$C_{\mathrm{A}}(\sigma, \ell_m) - cC_{\mathrm{OPT}}(\sigma, \ell_m) \geq m,$$

which proves the theorem. □

It is worthwhile noting that in [8] it was proved that under the every-step model, the above result is valid even if the adversary has a cache of size 1. The proof uses an adversary that does not serve the requests that are expensive for the on-line algorithm. That very strong result does not seem to be achievable under the other two models.

Extremely tight fairness restrictions. Note that Theorem 21 excludes an interesting case, that is, the case $t = w - 1$. In this case the situation is the same as in the finite version while all the sequences are active, so the discussion in Section 4.3 is valid here. As a consequence we have results analogous to those obtained for the finite problem. Under the every-step model a lower bound of k is straightforward, and our on-line algorithm RRFWF (Figure 5) is $(k + w)$-competitive.

THEOREM 22. *No on-line algorithm for Fair-IMTP is better than k-competitive under the every-step model, even if we restrict the sequences of requests to be formed by at most* $k + 1$ *distinct pages*.

PROOF. The proof is almost the same as Theorem 8, although with infinite sequences. □

THEOREM 23. *Algorithm RRFWF is* $(k+w)$*-competitive for Fair-IMTP with* $t = w-1$ *under the every-step model*.

PROOF. Suppose that after ℓ requests RRFWF is in the $(m + 1)$st super-phase. Then its cost is

$$C_{\mathrm{RRFWF}}(\sigma, \ell) \leq (k + w)m + k + w.$$

As in Theorem 11, at the end of each super-phase that RRFWF has completed the adversary must have served the same requests as RRFWF. Since in each super-phase at least $k+1$ distinct pages appear, the adversary must fault at least once in each completed super-phase, and then we have

$$C_{\mathrm{OPT}}(\sigma, \ell) \geq m,$$

so

$$C_{\mathrm{RRFWF}}(\sigma, \ell) \ \leq \ (k + w)C_{\mathrm{OPT}}(\sigma, \ell) + k + w. \qquad \square$$

We now extend the lower and upper bounds of the previous two theorems to cover Fair-IMTP with $t = w - 1$ under the three competitiveness models. We will prove the equivalence of the three models using the fact that in Fair-IMTP with $t = w - 1$, after each w requests any algorithm must have served the same set of requests.

THEOREM 24. *Consider the problem IMTP, and suppose that the algorithms are restricted to serving the requests row by row* (*as, for example, in Fair-IMTP with* $t = w-1$). *Then the following statements are equivalent*:

(I) *there exists a c-competitive on-line algorithm under the every-step model*;
(II) *there exists a c-competitive on-line algorithm under the constant-limit model*;
(III) *there exists a c-competitive on-line algorithm under the variable-limit model*.

PROOF. We saw in Theorems 12 and 13 that (I) $\Rightarrow$ (II) and that (II) $\Rightarrow$ (III), respectively. To finish the proof we will see that $\neg$(I) $\Rightarrow$ $\neg$(III). Let A be any on-line algorithm for the problem. We will show an input tuple σ such that $C_{\mathrm{A}}(\sigma, \ell) - cC_{\mathrm{OPT}}(\sigma, \ell)$ is not upper bounded. Let $D = 1 + c(2w - 1)$. Since $\neg$(I) holds, there exist σ and ℓ_1 such that

$$C_{\mathrm{A}}(\sigma, \ell_1) - cC_{\mathrm{OPT}}(\sigma, \ell_1) > D.$$

After ℓ_1 requests, A has not seen the requests that are after the row number $r_1 = \lceil \ell_1/w \rceil + 1$. To terminate that row the algorithms must serve at most $2w - 1$ new requests, and at the end of the row the costs will be

$$C_{\mathrm{A}}(\sigma, wr_1) \geq C_{\mathrm{A}}(\sigma, \ell_1)$$

and

$$C_{\mathrm{OPT}}(\sigma, wr_1) \leq C_{\mathrm{OPT}}(\sigma, \ell_1) + 2w - 1.$$

Therefore we have

$$\begin{aligned} C_{\mathrm{A}}(\sigma, wr_1) - cC_{\mathrm{OPT}}(\sigma, wr_1) &\geq C_{\mathrm{A}}(\sigma, \ell_1) - cC_{\mathrm{OPT}}(\sigma, \ell_1) - c(2w - 1) \\ &> D - c(2w - 1) = 1. \end{aligned}$$

From that point on, the adversary can cyclically start a new construction as in the beginning. In general, for any positive integer m there exist a number of requests ℓ_m and a row number $r_m = \lceil \ell_m/w \rceil + 1$ such that

$$C_{\mathrm{A}}(\sigma, wr_m) - cC_{\mathrm{OPT}}(\sigma, wr_m) > m. \qquad \square$$

In [17] it was proved that no on-line algorithm for Fair-IMTP with $t = w - 1$, even w, and $k = 1$ is better than $(k + w)$-competitive under the every-step model. Hence, by Theorem 24 it follows that no general on-line algorithm for Fair-IMTP outperforms RRFWF, under none of the three competitiveness models.

If we exclude the variable-limit model in Theorem 24, we are able to prove a stronger result.

THEOREM 25. *Let A be any on-line algorithm for IMTP, and suppose that the algorithms are restricted to serving the requests row by row* (*as, for example, in Fair-IMTP with* $t = w - 1$). *Then the following statements are equivalent*:

(I) *A is c-competitive under the every-step model*;
(II) *A is c-competitive under the constant-limit model*.

PROOF. From Theorem 12 we know that (I) $\Rightarrow$ (II), and then we only need to prove that $\neg$(I) $\Rightarrow$ $\neg$(II). Let D be any constant. Since $\neg$(I) holds, there exist σ and ℓ such that

$$C_{\mathrm{A}}(\sigma, \ell) - cC_{\mathrm{OPT}}(\sigma, \ell) > D + 1 + 2wc.$$

After ℓ requests, A has not seen the requests that are after the row number $r = \lceil \ell/w \rceil + 1$. The adversary has at most $2w - 1$ page faults to serve that row completely. After that, if

the same page is requested over and over again the adversary has at most one new page fault. Thus, the optimal off-line cost for any number of requests $\ell' \geq wr$ is

$$C_{\mathrm{OPT}}(\sigma, \ell') \leq C_{\mathrm{OPT}}(\sigma, \ell) + 2w.$$

The cost of A cannot decrease when the number of served requests grows, and then we have

$$\begin{aligned} C_{\mathrm{A}}(\sigma, \ell') - cC_{\mathrm{OPT}}(\sigma, \ell') &\geq C_{\mathrm{A}}(\sigma, \ell) - cC_{\mathrm{OPT}}(\sigma, \ell) - 2wc \\ &> (D + 1 + 2wc) - 2wc = D + 1, \end{aligned}$$

and so

$$\limsup_{\ell' \to \infty} C_{\mathrm{A}}(\sigma, \ell') \; - \; cC_{\mathrm{OPT}}(\sigma, \ell') \geq D + 1 > D. \qquad \square$$

6. Conclusions, Further Research, and Open Problems. In the same way that Paging has been a paradigmatic problem for traditional competitive analysis, the different variants of MTP that we discussed in this paper can serve as a first step toward establishing a general framework for multi-threaded on-line problems. New issues characteristic of this family of problems (mainly fairness and infinite inputs) have been deeply studied, deriving conclusions that may be generalized and extended to other problems. Our main goal is now to extend all our definitions to the multi-threaded versions of, for example, Metrical Task Systems; we already have some results in that direction [7].

Table 1 summarizes the competitiveness results presented in this paper. A conclusion that arises clearly from our work is that when fair behavior is imposed, things are much harder for on-line algorithms, to the extent that there are no competitive on-line algorithms in general. An interesting research direction is that of modeling fairness restrictions in a different way. One possibility is to strengthen the definition by considering (instead of t) an integer δ with the following meaning: the difference between the number

Table 1. Summary of our results.

	Unfair-FMTP		Unfair-IMTP	
	Lower bound	Upper bound	Lower bound	Upper bound
Any w	$k + 1 - 1/w$ [a]	wk	wk [b] $(0.5\lfloor \log w \rfloor + 1)k/e$ [c]	wk
	Fair-FMTP		Fair-IMTP	
	Lower bound	Upper bound	Lower bound	Upper bound
$t \geq w \geq 2$	∞	—	∞	—
$t = w - 1$ and $w \geq 3$	∞	—	k	$k + w$
$t = w - 1$ and $w = 2$	k	$k + w$	k	$k + w$

[a] A better lower bound of $(0.5\lfloor \log w \rfloor + 1)k/e$ is given in [6].
[b] Under the every-step and constant-limit models.
[c] Under the variable-limit model.

of served requests of any pair of sequences can never exceed δ. This or other models might allow the existence of general competitive on-line algorithms for the fair case.

On the other hand, the difference between finite and infinite inputs is not so significant from a competitive analysis point of view. The deep analysis we have done, considering three performance measures for the infinite case, shows only subtle differences among them (most of which can be observed in Table 1). Some of the differences could even disappear if, for example, a better lower bound is found for Unfair-FMTP.

As can be seen in Table 1, the lower and upper bounds do not coincide in every case, so one goal is to close the existing gaps. Finally, it would be interesting to extend the work done by Seiden in [15] on randomized algorithms for FMTP and IMTP under the every-step model to the other models, so as to analyze the relations existing among the competitiveness models for MTP in a randomized setting.

Acknowledgements. We thank Luis César Maiarú for useful discussions about this work. We are in debt to Steven S. Seiden for giving us ideas that led to a cleaner proof of Theorem 9.

References

[1] H. Alborzi, E. Torng, P. Uthaisombut, and S. Wagner. The k-client problem. In *Proceedings of the Eighth Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 73–82, New Orleans, Louisiana, 5–7 January 1997.

[2] L. A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5:78–101, 1966.

[3] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11:2–14, 1994.

[4] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, April 1995.

[5] A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the Association for Computing Machinery*, 39(4):745–763, October 1992.

[6] E. Feuerstein, D. G. Robak, and A. Strejilevich de Loma, 2000. Work in progress.

[7] E. Feuerstein, S. S. Seiden, and A. Strejilevich de Loma. On multi-threaded metrical task systems. Technical Report TR 99-008, Departamento de Computación, Universidad de Buenos Aires, November 1999. `http://www.dc.uba.ar`.

[8] E. Feuerstein and A. Strejilevich de Loma. On multi-threaded paging. In *Proceedings of the Seventh International Symposium on Algorithms and Computation* (*ISAAC* '96), pages 417–426, Volume 1178 of Lecture Notes in Computer Science. Osaka, Japan, 16–18 December 1996. Springer-Verlag, Berlin.

[9] E. Feuerstein and A. Strejilevich de Loma. Different competitiveness measures for infinite multi-threaded paging. Technical Report TR 98-021, Departamento de Computación, Universidad de Buenos Aires, November 1998. `http://www.dc.uba.ar`.

[10] A. Fiat and A. R. Karlin. Randomized and multipointer paging with locality of reference. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 626–634, Las Vegas, Nevada, 29 May–1 June 1995.

[11] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[12] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? Technical Report TR-92-044, ICSI, July 1992.

[13] T. Kimbrel. Interleaved prefetching. In *Proceedings of the Tenth Annual ACM–SIAM Symposium on Discrete Algorithms* (*SODA* '99), pages 557–565, Baltimore, Maryland, 17–19 January 1999.

[14] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. *IBM Journal of Research and Development*, 38(6):683–707, November 1994.

[15] S. S. Seiden. Randomized online multi-threaded paging. *Nordic Journal of Computing*, 6(2):148–161, 1999.

[16] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of ACM*, 28:202–208, 1985.

[17] A. Strejilevich de Loma. New results on fair multi-threaded paging. *Electronic Journal of SADIO*, 1(1):21–36, May 1998. `http://www.sadio.org.ar`.