Uniform Service Systems with k Servers*

Esteban Feuerstein

Depto. de Computación, FCEyN, Universidad de Buenos Aires
Instituto de Ciencias, Universidad de General Sarmiento, Argentina
efeuerst@dc.uba.ar

Abstract. We consider the problem of k servers situated on a uniform metric space that must serve a sequence of requests, where each request consists of a set of locations of the metric space and can be served by moving a server to *any* of the nodes of the set. The goal is to minimize the total distance traveled by the servers. This problem generalizes a problem presented by Chrobak and Larmore in [7]. We give lower and upper bounds on the competitive ratio achievable by on-line algorithms for this problem, and consider also interesting particular cases.

1 Introduction

During the last decade considerable attention has been devoted to competitive analysis of on-line algorithms. On-line problems have a variety of relevant applications in computer science, logistics, economy and robotics.

Probably the most famous on-line problem is the Paging Problem [15], that is the problem of managing a two-level memory, one level of limited capacity and fast access time (the cache) and the other one with slow access time but potentially unlimited capacity. An algorithm for this problem must determine which page of the cache to evict in front of a page-fault, with the goal of minimizing the total number of page faults incurred for serving a sequence of requests. On-line paging algorithms must decide which page to replace without knowledge of future requests.

On-line algorithms are in general evaluated using *competitive analysis* [11]: an on-line algorithm for a certain problem is said to be c-competitive if the cost incurred by it to serve *any* input is at most c times the cost charged to the optimal (off-line) algorithm for that input plus a constant.

One of the most challenging on-line problem is the k-server problem [12], in which k servers, located on a metric space, must serve a sequence of requests at points of the metric space. An on-line algorithm for that problem tries to minimize the total distance traveled by the servers, deciding which server is moved

* This work was partially supported by the KIT program of the European Community (Project DYNDATA), by University of Buenos Aires' Programación para Investigadores Jóvenes, project EX070/J "Algoritmos Eficientes para Problemas On-line con Aplicaciones" and by UBACYT project "Modelos y Técnicas de Optimización Combinatoria".

to each point in an on-line way. The great amount of extensions and generalizations of the Paging Problem include also the weighted version of Paging [14], the access graph model of [5], Metrical Task Systems [6] and Request-answer games [4].

In [7], Chrobak and Larmore proposed a family of on-line problems, namely Metrical Service Systems (MSS). In an instance of MSS_w *one* server situated on a metric space must serve a sequence of requests, where each request consists of a *set* of nodes of the metric space (of size at most w) and can be served by moving the server to *any* of the nodes of the set. The goal is to minimize the total distance traveled by the server. An important particular case of MSS is when the metric space is uniform, i.e. when all the distances are equal (uniform-MSS). Both MSS and uniform-MSS are particular cases of Metrical Task Systems, but not of the k-server problem, as each request specifies different alternative nodes to cover.

In this paper we present the generalization of uniform-MSS_w to the case in which $k \geq 1$ servers are used. We call this problem (k, w)-Uniform Service Systems (abbreviated as $\text{USS}_{(k,w)}$).

It is a well known fact that the k-server problem on uniform metric spaces is isomorphic to the paging problem with a cache of size k. Analogously, $\text{USS}_{(k,w)}$ can be seen as the following generalization of the Paging problem: given a set U of pages, an on-line algorithm with a cache of size k must deal with a finite sequence of requests, each of which consists in a subset $r \subset U$ of size at most w. Each request is served by having in the cache *at least* one element of r. In the reminder of this paper we shall use this "Paging-oriented" terminology rather than that of server problems.

The problem, in both its server and paging versions, has several natural applications. As an example, consider a distributed network with virtual-circuit routing, in which each processor may have at most a constant number of simultaneously enabled connections. If the data needed to perform some task are replicated over the network, a processor has the alternative of communicating with different processors being forced, in general, to close its connection with some other one. Another example is given by a k mobile servers that can give their service in any branch of each of the clients' companies. The decision about where to serve each request will influence the total time needed to process a sequence of requests, as well as the decision of which server to assign to each request.

The problem treated in this paper is a sort of "dual" of the one considered in [8], in which every request consists of a set of pages, all of which have to be present in the cache to serve the request.

Other related work has been done by Ausiello et al. [3, 2], Alborzi et al. [1] and by Feuerstein et al. [9]. In [3, 2] the problem of efficiently serving a sequence of requests in a metric space presented in an on-line fashion is considered. At every moment, a server may decide which of the requests to serve, with the goal of minimizing the total completion time. A similar approach is taken in [1] where it is assumed that a *fixed* number of clients present sequences of requests in a

metric space, that must be served by a single server. At any time, each client has at most one request to be served, after which a new one may be presented. They consider different cost models, namely the make-span, total completion time and maximum response time. The main difference with the previously cited works is that, as the requests are threaded, the time in which requests are presented depends on the order in which the server processes previous requests. Finally, [9] introduces the generalization of Paging to the case where there are many threads of requests. That 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. The difference with the approach taken in this paper is that, in the setting of [9], all the requests that are not served at some stage are repeated in the next one, while here a brand-new set of pages may be requested.

We show that no on-line algorithm for $\mathrm{USS}_{(k,w)}$ can achieve a competitive ratio better than $\binom{k+w}{w} - 1$, and we present an $O(k \min(k^w, w^k))$-competitive algorithm. For any fixed value of k (and arbitrary w) this is at most a constant factor away from the lower bound . However, for k tending to infinity the upper bound is a constant times k away from optimality. We conjecture that the same algorithm achieves a competitive ratio of $O(\min(k^w, w^k))$, and therefore obtains an optimal (up to a constant factor) competitive ratio also when k tends to infinity, but we have not proved it in the general case. However, we have proved it for $w = 2$ when the requests verify certain restrictions that will be explained later. Our algorithm solves at each step an instance of an NP-complete problem. In Section 4 we present a polynomial-time algorithm that achieves a competitive ratio of $2k$ with a cache of size $2k$ against an adversary with a cache of size k. The lower and upper bounds we obtain generalize the results of [7] regarding uniform metric spaces.

2 The General Case

The following is a lower bound on the competitive ratio of any on-line algorithm for this problem.

Theorem 1. *If* $c < \binom{k+w}{w} - 1$ *then no on-line algorithm for* $USS_{(k,w)}$ *is* c*-competitive.*

Proof. We consider a universe U of cardinality $k+w$. Given an on-line algorithm A, we construct a sequence σ_A in the following way: each request of σ_A is to the w pages that are not present in A's cache, and hence A faults at each request. We can assume that A evicts only one page at each fault, otherwise we could construct an algorithm A' behaving in that way and such that for every sequence σ, $C_{A'}(\sigma) \leq C_A(\sigma)$ (for any algorithm ALG, $C_{ALG}(\sigma)$ denotes the cost incurred by ALG to serve a sequence σ of requests, the sequence is omitted when it is clear from the context). Hence, if we let $|\sigma_A| = n$, we have $C_A = n$. Consider now the family $\mathcal{F}$ of all the subsets of U of cardinality w. We have that $|\mathcal{F}| = \binom{k+w}{w}$.

At any moment, the configuration of the cache of any algorithm (off-line or on-line) can be associated to the set f of the pages of U that are not present in the algorithm's cache, with $f \in \mathcal{F}$. Consider a family of adversaries ADV_i such that each adversary is in one of the possible $\binom{k+w}{w}$ configurations, an exception made of the configuration of A. There are $\binom{k+w}{w} - 1$ such adversaries, and all of them can serve each request r of σ_A with no cost. Now assume that A evicts some page $x \notin r$ to make place for a page $y \in r$. Then, the only adversary whose cache would coincide with A's cache *after* A's move evicts y and brings x; but only after serving the request r. Then we can think as all adversaries together paying a cost of 1 for each request. Hence, we have that $\sum_i C_{ADV_i} = n$. As there are in total $\binom{k+w}{w} - 1$ adversaries, the average cost is $\frac{n}{\binom{k+w}{w}-1}$. But hence there is at least one i such that C_{ADV_i} is not more than average, and the ratio $\frac{C_A}{C_{ADV_i}}$ is at least $\binom{k+w}{w} - 1$. □

In the following we propose an algorithm for $\mathrm{USS}_{(k,w)}$. The algorithm, which we call the *Hitting Set algorithm* (HS) is $k \min(\frac{k^{w+1}-k}{k-1}, \sum_{i=0}^{k-2} w^i + w^k)$-competitive. Notice that $\sum_{i=0}^{k-2} w^i + w^k \leq 2w^k$.

HS divides the sequence of requests in phases, the first phase starting with the first request of the sequence. Each time HS cannot answer the current request it behaves as follows. First, it computes a minimum cardinality set H of pages that intersects all the requests that produced a fault during the current phase. If $|H| \leq k$ then H is brought into the cache. If there is more than one set with minimum cardinality, then the one that can be brought into the cache with minimum cost is chosen. Otherwise, if $|H| > k$ the current phase is finished and a new phase starts with the present request.

The constraint of choosing the next configuration of the cache so as to minimize the (Hamming) distance to the current one is not really used in the proof of the following theorem, but it is a necessary condition to prove the better bound of $O(k^w)$ that we conjecture is achieved by the algorithm. Notice that the "lazy" version of HS, that instead of bringing into the cache a complete hitting set H brings just one page of H useful to serve the current request, can be forced to pay as much as HS, simply by repeating, after each "new" request, all the previous requests of the phase.

Theorem 2. *HS is* $k \min(\frac{k^{w+1}-k}{k-1}, \sum_{i=0}^{k-2} w^i + w^k)$*-competitive for* $USS_{(k,w)}$.

Proof. By definition of HS, all the requests of a phase plus the first one of the following phase need at least $k+1$ different pages to be answered. Therefore, we know that the adversary pays at least a cost of 1 for each phase. We will show that HS may fault on at most $\min(\frac{k^{w+1}-k}{k-1}, \sum_{i=0}^{k-2} w^i + w^k)$ different requests during a phase. By definition during a phase HS does not fault twice on the same request, and the cost for each single request is at most k, therefore the total cost of HS during each phase is not more than k times the number of faults, and the thesis follows. We will separately prove that:

i) the maximum number of faults during a phase is at most than $\sum_{i=1}^{w} k^i = \frac{k^{w+1}-k}{k-1}$, and ii) it is at most than $\sum_{i=0}^{k-2} w^i + w^k$.

i) Let us first see that the maximum number of faults of a phase can be at most $\sum_{i=1}^{w} k^i$. We will proceed by induction on w.

Basis ($w = 2$). Let H be the final hitting set of a phase, i.e. the last set of cardinality not greater than k such that all the requests of the phase could be served by having H in the cache. Let $v \in H$. After at most $k+1$ *different* requests including v, it is clear that any hitting set of cardinality less than $k+1$ contains v. Hence HS may fault at most $k+1$ times for each of the at most k pages in H. Summing over all pages of H we get that the number of different requests of a phase may not exceed $k^2 + k$.

Inductive step. Suppose that the thesis is true for $w \leq \ell - 1$, we will prove it for $w = \ell$. Consider the family $\mathcal{F}_v$ of all the requests containing a particular page v and the family $\mathcal{F}'_v = \{F - \{v\} | F \in \mathcal{F}_v\}$ of the requests of $\mathcal{F}_v$ *without* v. Obviously, for every $F \in \mathcal{F}'_v$ we have that $|F| \leq \ell - 1$, and every hitting set of $\mathcal{F}_v$ not containing v must be a hitting set of $\mathcal{F}'_v$. By inductive hypothesis, after at most $1 + \sum_{i=1}^{\ell-1} k^i$ requests of this type every hitting set of $\mathcal{F}'_v$ must be of cardinality at least $k+1$, and hence HS will necessarily keep v to "cover" $\mathcal{F}_v$ until the phase is over. Therefore, summing over all pages of the final hitting set we get that the maximum number of faults in a phase is at most $k(1 + \sum_{i=1}^{\ell-1} k^i) = k + \sum_{i=2}^{\ell} k^i = \sum_{i=1}^{w} k^i$.

ii) We will now prove, by induction on k, that the maximum possible number of faults in a phase is $\sum_{i=0}^{k-2} w^i + w^k$.

Basis ($k = 1$). The first request of the phase gives at most w alternatives, and hence there are at most w different configurations in which that request can be served.

Inductive step. We assume the thesis is true for $k \leq \ell - 1$ and will prove it for $k = \ell$. Consider the first request r such that the size of the minimum hitting set of all the requests so far in the phase becomes ℓ. That request can be covered in at most w different ways. By inductive hypothesis, for each of these ways there may be at most $\sum_{i=0}^{\ell-3} w^i + w^{\ell-1}$ requests before the other $\ell - 1$ elements of the hitting set become fixed. Then the total number of faults during a phase can be at most $1 + w(\sum_{i=0}^{\ell-3} w^i + w^{\ell-1}) = 1 + \sum_{i=1}^{\ell-2} w^i + w^\ell = \sum_{i=0}^{\ell-2} w^i + w^\ell$. □

From the previous Theorem we get the following corollary regarding the performance of the algorithm HS for MSS_w on uniform metric spaces (MSS_w is the class of Metrical Service Systems where all requested sets have cardinality at most w). Our bound coincides with that of [7] for this particular case.

Corollary 3. *HS is w-competitive for MSS_w on uniform metric spaces.* □

To conclude the analysis of the performance of HS for $\text{USS}_{(k,w)}$, we will quantify the gap between the upper bound of Theorem 2 and the lower bound of Theorem 1 in the two limit cases, i.e. when either w or k tend to infinity and the other value is fixed.

If we denote as C the competitive ratio achieved by algorithm HS (that is, $C = k\min(\frac{k^{w+1}-k}{k-1}, \sum_{i=0}^{k-2} w^i + w^k)$), we have that for any fixed k, there is a value $w(k)$ such that $\forall w \geq w(k), C = k(\sum_{i=0}^{k-2} w^i + w^k)$. Conversely, for any fixed w there exists a value $k(w)$ such that $\forall k \geq k(w), C = k\frac{k^{w+1}-k}{k-1}$. Applying Stirling's formula we get the following values for the ratio $C/\binom{k+w}{w}$ (recall that $\sum_{i=0}^{k-2} w^i + w^k \leq 2w^k$):

$$\lim_{w\to\infty} \frac{C}{\binom{k+w}{w}} \leq \frac{k^{k+1}\sqrt{k}2\sqrt{2\pi}}{e^k} ,$$

$$\lim_{k\to\infty} \frac{C}{\binom{k+w}{w}} = \lim_{k\to\infty} \frac{\sqrt{2\pi}\sqrt{w}w^w k}{e^w} .$$

Note that in the first case the ratio tends to a constant, while in the second case it grows proportionally to the value of k.

3 The Acyclic Case for $w = 2$

In the particular case of requests of cardinality 2, every request can be seen as an edge of a graph with nodes the universe U, and any set of requests can be seen as a graph on U. In this case algorithm HS reduces to computing a minimum vertex cover of the subgraph determined by the requests of the current phase. In the following we will show that if the graph of the requests is acyclic, then HS obtains a competitive ratio that is at most a factor of 2 away from optimality. Actually, to prove this we need a smoother restriction on the input sequence, we only need that the (at most) $k^2 + k$ requests that form each phase do not form cycles. This particular case of $\text{USS}_{(k,2)}$ is called *acyclic*-$\text{USS}_{(k,2)}$. We need some definitions and preliminary results before stating and proving the theorem.

Definition 4. Given graph G and a node n of G, we say that n is:

- *fixed* if $n \in H$ for every minimum cardinality vertex cover H of G;
- *free* if there exist two minimum cardinality vertex covers H and H' of G such that $n \in H$ and $n \notin H'$;
- *forbidden* if there is no minimum cardinality vertex cover H of G such that $n \in H$.

Making some abuse of notation, we will see the pages of U as the nodes of the graph induced by the requests of the phase. Besides, we will continue referring to Hitting Sets instead of Vertex Covers.

Lemma 5. *After a request involving a free and a forbidden node, the number of free nodes in HS's cache is decremented exactly by the cost paid by HS.*

Proof. Suppose the request is to the pair $\{x, y\}$, where x is free and y is forbidden. Then, after the request, x will be part of every minimum hitting set at least until the size of a minimum hitting set is incremented, and by definition of HS this request is served by bringing x into the cache, possibly together with some other nodes. In general, HS will replace a set Z of free nodes by a set X, with $x \in X$ and $|Z| = |X|$. We will proceed by induction on the cardinality of X and Z.

Basis. If HS replaces some node z by x, the cost is 1, and the number of free nodes in the cache is decremented by 1, because z was free and x is now fixed.

Induction. We will consider the following two cases:

(1) There is only one neighbor z of x with $z \in Z$. Then we have two possibilities:
   (a) All the neighbors of z (except x) where already in the cache. Then there is a minimum Hitting Set obtained replacing z by x, a contradiction because we are supposing that the minimum change is of cardinality bigger than one.
   (b) Some neighbors $\{v_1, \ldots, v_j\}$ of z must be brought to the cache so as to cover the edges (z, v_i) that remain uncovered because of the eviction of z. The changes done to bring each v_i into the cache are the same that would have been done if a request $\{v_i, w\}$ had been requested, for some forbidden node w. By induction, all the nodes brought in such cases remain fixed, and hence the same happens in this case. Therefore, all the nodes brought in this case are fixed.

(2) There are $j > 1$ neighbors $z_1, \ldots, z_j$ of x with $z_i \in Z, i = 1, \ldots, j$. Since we are considering the acyclic case, all the requests of the phase form a forest. Let T be the subtree of that forest induced by $Z \cup X$. Let $T_i, i = 1, \ldots, j$ be the subtrees of T rooted at $z_1, \ldots, z_j$ respectively, and define $X_i = X \cap T_i$ and $Z_i = Z \cap T_i$ for $i = 1, \ldots, j$. As $|Z| = |X|$, and because $x \in X$, we have that $|X - \{x\}| = |Z| - 1$. Note that $X - \{x\} = \bigcup_{i=1}^{j} X_i$. Then there is at least one i such that $|X_i| < |Z_i|$. But in this case replacing only Z_i by $X_i \cup \{x\}$ would give a hitting set of the same or smaller cardinality than the previous hitting set, contradicting the hypothesis that replacing Z by X was a minimum cost change (or contradicting the minimality hypothesis of the previous hitting set). □

Theorem 6. *HS is $O(k^2)$-competitive for acyclic-$USS_{(k,2)}$.*

Proof. We already know that the adversary incurs at least in a cost of 1 during each phase, and we will bound HS's cost during the phase. For this we will use the following potential function:

$$\Phi = (k+2)|H| - |f| - k|t|$$

where H is a minimum hitting set, f is the set of free nodes in HS's cache and t is the number of non-trivial connected components (trees), all referred to the graph induced by the requests of the current phase that produced a fault.

The value of Φ is 1 after the first request of a phase, and is never greater than $k^2 + 2k$. Therefore, if we show that Φ is incremented at least by the cost paid by HS for each move we have that the total cost charged to HS during the phase is not greater than $k^2 + 2k$. We will now see that this holds.

The first thing to note is that requests involving at least one fixed node are served by HS without cost (as by definition of HS fixed nodes are present in the cache). Those requests are ignored by the algorithm, and produce no variation in the potential function. Hence we have to analyze the variation of Φ in the remaining cases, that are:

Forbidden/Forbidden: In this case, the cardinality of a minimum hitting set is incremented by one, and the cost paid by HS is exactly 1. We can distinguish two sub-cases:
- The requested nodes are both trivial trees, and hence the number of free nodes and the number of non-trivial trees increase by 1. Then we have that $\Delta\Phi = (k+2)(|H|+1)-(k+2)|H|-1-k(t+1)+kt = k+2-1-k = 1$.
- At least one of the requested nodes was part of some non-trivial tree. Then the number of non trivial connected components does not increase, and the number of free nodes may increase at most by $|H| + 1$. Hence $\Delta\Phi \geq (k+2)(|H|+1) - (k+2)|H| - |H| - 1 = k + 2 - |H| - 1 \geq 1$.

Forbidden/Free: In this case, the size of the minimum hitting set remains unchanged, and the number of non-trivial connected components does not increase. By Lemma 5, the decrement in the number of free nodes in cache is equal to the cost paid by HS, yielding an increment in Φ equal to that cost.

Free/Free: The size of the minimum hitting set and the number of free nodes in cache do not change. As for the number of non trivial connected components, it must necessarily decrease by 1, and therefore we have $\Delta\Phi = -k(t-1) + kt = k$. But k is always greater than the cost paid by HS. Note that here we use the fact that the graph of the requests of a phase is acyclic. □

4 A Polynomial-time Algorithm

HS has an important drawback regarding its efficiency. In fact, at each fault HS has to solve an instance of the Hitting Set problem, which is NP-complete, even in the case of $w = 2$ [10]. We propose a polynomial-time on-line algorithm that with a cache of size $2k$ is $2k$-competitive against adversaries with cache size k for $\text{USS}_{(k,2)}$. The algorithm (and its analysis) is based on the ideas of a well-known 2-approximate polynomial algorithm for the Minimum Vertex Cover problem (see for example [13]). We call this algorithm AHS, and it works as follows: The sequence of requests is divided in phases, each phase ending after the k-th fault of the algorithm. A request r is answered in the following way: if at least one of the pages of r is present in the cache, then nothing is done. Otherwise *both* pages of r are brought into the cache and marked, evicting 2 unmarked pages. If there are no unmarked pages to evict (i.e. there have already been k faults in the current phase), then all pages are unmarked and the phase is over.

AHS follows in a certain way the same philosophy of HS, but keeping a 2-approximate hitting set of the requests of the phase instead of a minimum hitting set. Because of this fact, it needs a memory twice as big as that of the adversary to prevent infinite sequences of requests that could be served by the adversary with constant cost and that could otherwise produce an unbounded cost each time the optimal hitting set does not coincide with the one computed by AHS.

Theorem 7. *Algorithm AHS with cache-size $2k$ is $2k$-competitive against adversaries with cache-size k for $USS_{(k,2)}$.*

Proof. Trivially the total cost of AHS during a phase is $2k$. On the other hand, consider the k requests that produced a fault during a phase plus the first request of the following phase. By definition of AHS, these requests are $k+1$ pairwise disjoint sets, and hence they need at least $k+1$ different pages to be answered. So we conclude that the adversary must necessarily have at least one fault to serve all of them. □

Algorithm AHS can be naturally extended to deal with requests of size at most w, and almost the same proof of Theorem 7 holds for the following Theorem. The idea is that a w-approximate solution to the Minimum Hitting Set problem with sets of cardinality at most w may be obtained by an algorithm that considers the sets one by one and, if the current set is uncovered then it adds all its elements to the hitting set. This corresponds to extending the approximate solution of Minimum Vertex Cover of graphs based on a maximal matching to an approximate solution of Minimum Vertex Cover of w-hypergraphs based on the computation of a maximal w-dimensional matching.

Theorem 8. *Algorithm AHS with cache-size wk is wk-competitive against adversaries with cache-size k for $USS_{(k,w)}$.* □

5 Open Problems and Future Research

The main open problem is to close the gap between the lower bound of Theorem 1 and the upper bound of Theorem 2. As we stated before, we believe that our algorithm HS achieves better performance than what has been proved.

An interesting subject of future research is to extend $USS_{(k,w)}$ to non-uniform metric spaces. This would extend both the work in this paper and the work by Chrobak and Larmore [7] on Metrical Service Systems, were only one server is considered.

Acknowledgments. I am very grateful to Amos Fiat and Alberto Marchetti-Spaccamela for useful discussions during early stages of this work.

References

1. H. Alborzi, E. Torng, P. Uthaisombut, and S. Wagner. The k-client problem. In *Proc. of Eigth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1995.
2. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Competitive algorithms for the traveling salesman. In *Proc. of Workshop on Algorithms and Data Structures (WADS'95)*, Springer-Verlag, 1995.
3. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Serving requests with on-line routing. In *Proc. of 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)*, pages 37–48, Springer-Verlag, July 1995.
4. S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Widgerson. On the power of randomization in on-line algorithms. *Algorithmica*, 11:2–14, 1994.
5. A. Borodin, Sandy Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. In *Proc. of 23rd ACM Symposium on Theory of Computing*, pages 249–259, 1991.
6. A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task system. *Journal of the Association for Computing Machinery*, 39(4):745–763, 1992.
7. M. Chrobak and L. Larmore. The server problem and on-line games. In *On-line Algorithms*, pages 11–64, AMS-ACM, 1992.
8. E. Feuerstein. Paging more than one page. In *Proceedings of the Second Latin American Symposium on Theoretical Informatics (LATIN95)*, pages 272–287, Springer-Verlag, 1995. An improved version of this paper will appear in *Theoretical Computer Science* (1997).
9. E. Feuerstein and A. Strejilevich de Loma. On multi-threaded paging. In *Proceedings of the 7th International Symposium on Algorithms and Computation (ISAAC'96)*, Springer-Verlag, 1996.
10. M. R. Garey and D. S. Johnson. *Computers and Intractabiliy - A Guide to the Theory of NP-completeness.* W.H. Freeman and Company, San Francisco, 1979.
11. A. Karlin, M. Manasse, L. Rudolph, and D. Sleator. Competitive snoopy caching. *Algorithmica*, 3():79–119, 1988.
12. M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.
13. R. Motwani. *Lecture Notes on Approximation Algorithms.* Technical Report, Stanford University.
14. P. Raghavan and M. Snir. *Memory versus randomization in on-line algorithms.* RC 15622, IBM, 1990.
15. D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of ACM*, 28(2):202–208, 1985.