On Space Bounded Server Algorithms

Ganesh R. Baliga

Department of CIS
University of Delaware
Newark, DE 19716
(email: baliga@udel.edu)

Anil M. Shende

Department of CS
Bucknell University
Lewisburg, PA 17837
(email: shende@bucknell.edu)

Abstract

This paper is motivated by the need for fast algorithms for on-line problems, many of which require algorithms to provide solutions in real-time. In this paper, we study a class of arguably fast algorithms (called space bounded algorithms) for the k-server problem on finite metric spaces. Specifically, we provide a necessary and sufficient condition for such algorithms to be competitive on finite metric spaces. Moreover, this characterization provides a strategy for designing competitive algorithms *for finite metric spaces.*

1 Introduction

On-line problems arise quite often, and quite naturally in computer science. The dynamic nature of these problems demands that algorithms for these problems make decisions without full knowledge of the impact of these decisions on the future performance of the algorithms. Many data structuring problems and scheduling problems are on-line problems. Server problems [7] are a variation of the generalized formulation of on-line problems [2]. Server problems can be used to model a variety of problems in computer science including, among others, paging and planning the motion of the several heads in a multi-headed disk. Informally, the k-server problem requires the design of an algorithm, that uses k servers, and that, given as input a point (typically called a *request*) in a metric space, schedules at least one of its k servers to move to that point, and then waits for the next request. Moving a server to a request point is typically called *servicing the request.* Furthermore, it is desired that the cumulative distance moved by all of the algorithms' k-servers in servicing a sequence of requests (also referred to as the *cost* incurred by the algorithm) be as small as possible. Those k-server algorithms that, for all request sequences, have a cost within a constant factor of the optimal off-line cost are called *competitive*, and the factor is called the *competitive factor* of the algorithm.

Most real-life problems that have been modeled as k-server problems receive requests from a finite metric space. Moreover, many such problems (e.g., motion planning of the heads of a multi-headed disk) require algorithms which need to effectively deal with extremely large request sequences within miniscule time frames. Thus, algorithms for such problems need to provide good solutions in *real-time.* We can argue, then, that algorithms that require arbitrarily large unbounded amounts of scratch space, e.g., algorithms that physically store and use information about *all* the past requests to decide how to service the current request, could not possibly schedule servicings in real-time, thus rendering them impractical for such on-line problems. Motivated by the above considerations, we restrict ourselves to the k-server problem on finite metric spaces and focus our attention on the class of k-server algorithms that use a constant amount of scratch space (memory). We call these algorithms *space bounded* (see Section 3).

It is conjectured [7] that for each k and every metric space, there is a k-server algorithm that has a competitive factor k; this is now known as the *k-server conjecture.* This conjecture is known to hold for the 2 server case [4, 6]. Also, the conjecture is known to hold (for arbitrary k) for some special kinds of metric spaces such as lines and trees [3, 5]; in fact, the witnessing algorithms are *memoryless*. In contrast, it is known that there is *no* memoryless optimally competitive 2-server algorithm on the Euclidean plane [1]. Memoryless algorithms are a special case of the space bounded algorithms that are discussed in this paper.

For finite metric spaces with n points, (optimally) competitive algorithms are known only for 2 and $(n-1)$ servers [7], leaving open the search for competitive k-server algorithms for $2 < k < n-1$. We settle this question via our characterization of the set of space bounded algorithms that are competitive in a given finite metric space. In fact, our result yields a method for effectively deciding if a given space bounded algorithm is competitive in a given finite metric space, and also a method for designing fast, competitive algorithms for any given space bound and any finite metric space!

For completeness, we first formally define the k-server problem and some related notions in Section 2.

2 The k-server Problem

Let $\mathcal{M}$ be a metric space with metric $d_{\mathcal{M}}(\cdot,\cdot)$. We conceptualize $\mathcal{M}$ as being a set of points $r_1, r_2, \ldots, r_i, \ldots$ Finite metric spaces are those which contain only finitely many points. For a finite metric space $\mathcal{M}$, the diameter of $\mathcal{M}$ is equal to $\max\{d_{\mathcal{M}}(r_i, r_j) \mid r_i, r_j \in \mathcal{M}\}$. We use σ and σ' to denote *request sequences*, i.e., strings in $\mathcal{M}^*$. For all σ and finite σ', $\sigma' \cdot \sigma$ will denote the request sequence σ' followed by requests in σ. For each $i \geq 1$, σ^i denotes the request sequence σ repeated i times.

2.1 Definitions

Definition 1

1. A *k-configuration in $\mathcal{M}$* is a multiset of cardinality k and comprising of points in $\mathcal{M}$. $\mathcal{C}^k_{\mathcal{M}}$ denotes the set of all k-configurations in $\mathcal{M}$.

2. A *matching* f_{c_1,c_2} between k-configurations $c_1, c_2 \in \mathcal{C}^k_{\mathcal{M}}$ is a multiset bijection mapping points in c_1 to points in c_2; the *weight* of f_{c_1,c_2}, denoted by *weight*(f_{c_1,c_2}), is equal to $\sum_{i \in c_1} d_{\mathcal{M}}(i, f_{c_1,c_2}(i))$. For all $c_1, c_2 \in \mathcal{C}^k_{\mathcal{M}}$, $D_{\mathcal{M}}(c_1, c_2)$ denotes the *distance between configurations c_1 and c_2* and is equal to the weight of the matching between the two configurations with minimum weight.

3. A sequence $c_0 c_1 c_2 \ldots$, such that for all $i \geq 1, c_i \in \mathcal{C}^k_{\mathcal{M}}$ is a *k-service in $\mathcal{M}$*. We will also say that this k-service in $\mathcal{M}$ starts at c_0. The *cost of a k-service $s = c_0 c_1 c_2 \ldots$ in $\mathcal{M}$*, denoted $\mathtt{cost}_{\mathcal{M}}(s)$, is given by

$$\mathtt{cost}_{\mathcal{M}}(s) = D_{\mathcal{M}}(c_0, c_1) + D_{\mathcal{M}}(c_1, c_2) + \cdots.$$

   Let $\sigma = r_1 r_2 \ldots$ be a request sequence in $\mathcal{M}$. A k-service in $\mathcal{M}$, $c_0 c_1 c_2 \ldots$, is a *k-service for σ starting at c_0* iff for all $i \geq 1, r_i \in c_i$. For all $c \in \mathcal{C}^k_{\mathcal{M}}$, for all σ, $s^k_{\mathcal{M}}(c,\sigma)$ denotes the set of all possible k-services for σ starting at c in $\mathcal{M}$.

4. For all $c \in \mathcal{C}^k_{\mathcal{M}}$, for all σ, the *optimal cost in $\mathcal{M}$ for servicing σ starting at c with k servers*, denoted $\mathtt{opt}^k_{\mathcal{M}}(c,\sigma)$, is the minimum cost over all k-services in $s^k_{\mathcal{M}}(c,\sigma)$.

We now proceed to formally define a k-server algorithm and the notion of competitiveness of such algorithms.

Definition 2

1. $\mathcal{A}$ is a *k-server algorithm in $\mathcal{M}$* iff $[\mathcal{A} : \mathcal{C}^k_{\mathcal{M}} \times \mathcal{M}^* \to \mathcal{C}^k_{\mathcal{M}}]$ and $(\forall c \in \mathcal{C}^k_{\mathcal{M}})(\forall \sigma)(\forall r \in \mathcal{M})[r \in \mathcal{A}(c, \sigma \cdot r)$ and $\mathcal{A}(c, \sigma \cdot r) = \mathcal{A}(\mathcal{A}(c,\sigma), r)]$.

2. Suppose $\mathcal{A}$ is a k-server algorithm. Then, for all request sequences $\sigma = r_1 r_2 \ldots$ and for all k-configurations $c \in \mathcal{C}^k_{\mathcal{M}}$, $\overline{\mathcal{A}(c,\sigma)}$ is the sequence of configurations starting at c, given by $\overline{\mathcal{A}(c,\sigma)} = c \cdot \mathcal{A}(c, r_1) \cdot \mathcal{A}(c, r_1 r_2) \ldots \mathcal{A}(c, r_1 r_2 \ldots r_i) \ldots$ The *cost incurred by $\mathcal{A}$ in servicing σ, starting at c, in $\mathcal{M}$*, denoted $\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, c, \sigma)$, is given by

$$\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, c, \sigma) = \mathtt{cost}_{\mathcal{M}}(\overline{\mathcal{A}_{\mathcal{M}}(c,\sigma)}).$$

3. For $\mathcal{B}$, a real number, a k-server algorithm is *$\mathcal{B}$-competitive in a metric space $\mathcal{M}$* iff there exists a real number b such that for all $c \in \mathcal{C}^k_{\mathcal{M}}$ and for all σ,

$$\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, c, \sigma) \leq \mathcal{B} \cdot \mathtt{opt}^k_{\mathcal{M}}(c,\sigma) + b.$$

   A k-server algorithm in $\mathcal{M}$ is said to be *competitive in $\mathcal{M}$* iff it is $\mathcal{B}$-competitive in $\mathcal{M}$ for some $\mathcal{B}$.

We now define the notion of a ρ-distinct sequence in $\mathcal{M}$, and then, in Lemma 1, isolate an intuitive property of optimal k-services for request sequences in a large class of metric spaces. Lemma 1 applies to metric spaces $\mathcal{M}$ for which there exists a $\mu > 0$ such that

$$\min\{d_{\mathcal{M}}(r_i, r_j) \mid r_i, r_j \in \mathcal{M}\} > \mu.$$

Finite metric spaces are clearly a proper subclass of such metric spaces. The following notion is necessary for stating Lemma 1 an useful in our later exposition.

Definition 3 A request sequence $r_1 r_2 \ldots$ in $\mathcal{M}$ is a *ρ-distinct sequence in $\mathcal{M}$* iff the cardinality of the set of distinct points in the sequence is exactly ρ.

We state Lemma 1 without proof; the proof follows from an application of the pigeon-hole principle.

Lemma 1 *Suppose $\mathcal{M}$ is a metric space wherein the distance between any two distinct points is at least $\mu > 0$. Then, for every $k \geq 1$, for every $c \in \mathcal{C}^k_{\mathcal{M}}$, for every finite σ and every σ', a $(k+1)$-distinct sequence, $\mathtt{opt}^k_{\mathcal{M}}(c, \sigma.\sigma') \geq \mathtt{opt}^k_{\mathcal{M}}(c, \sigma) + \mu$.* □

3 Space Bounded k-Server Algorithms

In this section, we study k-server algorithms that use a bounded amount of scratch space for their computations; we call such algorithms *space bounded* algorithms. Furthermore, we restrict ourselves to requests from finite metric spaces. Examples of space bounded algorithms are algorithms which perform bounded look back on the request sequence.

We first formally define the notion of space bounded algorithms. Then we define certain technical notions for space bounded k-server algorithms working on requests from a finite metric space. Using these notions we characterize space bounded algorithms that are competitive in finite metric spaces (see Theorem 1 below). We end this section with two important consequences of this characterization (see Theorems 2 and 3).

Definition 4 An algorithm $\mathcal{A}$ is *I-space bounded* iff on any input, $\mathcal{A}$ uses at most I bits of memory in its computation. An algorithm $\mathcal{A}$ is *space bounded* iff it is I-space bounded for some I.

Let $\mathcal{A}$ be an I-space bounded k-server algorithm on a finite metric space $\mathcal{M}$. (Note that the value of I is a property of $\mathcal{A}$, and is independent of any particular request sequence that is being processed by $\mathcal{A}$.) Prior to servicing a request, $\mathcal{A}$'s state can be effectively captured by the current positions of $\mathcal{A}$'s k servers along with the settings of the I memory bits used by $\mathcal{A}$. Thus $\mathcal{A}$ can in fact be in one of $\mathcal{C}^k_{\mathcal{M}} \times 2^I$ states. $\mathcal{A}$ will service a request based on its current state and the position of the request in $\mathcal{M}$.

Definition 5

1. Suppose $\mathcal{A}$ is an I-space bounded k-server algorithm. Suppose $\mathcal{M}$ is a finite metric space. The *graph induced by $\mathcal{A}$ on $\mathcal{M}$*, denoted $G_{\mathcal{A}}(\mathcal{M})$, is the weighted, directed multi-graph with labeled edges and with vertex set $(\mathcal{C}^k_{\mathcal{M}} \times 2^I)$. There is a directed edge $((c_1, \iota_1), (c_2, \iota_2))$ labeled r in $G_{\mathcal{A}}(\mathcal{M})$ iff there is an $r \in \mathcal{M}$ such that $f_{\mathcal{A}}((c_1, \iota_1), r) = (c_2, \iota_2)$. Furthermore, this edge is assigned a weight equal to $D_{\mathcal{M}}(c_1, c_2)$. (**Note**: All edges between two vertices have the same weight. Furthermore, for any c in $\mathcal{C}^k_{\mathcal{M}}$ if there is an edge $((c, \iota_1), (c, \iota_2))$ in $G_{\mathcal{A}}(\mathcal{M})$, then its weight is 0.) Vertices of $G_{\mathcal{A}}(\mathcal{M})$ are called *states induced by $\mathcal{A}$ on $\mathcal{M}$*, and the set of vertices, denoted by $\mathcal{S}_{\mathcal{A}}(\mathcal{M})$, is called the *state space of $\mathcal{A}$ in $\mathcal{M}$*. For any two states s_1, s_2 in $\mathcal{S}_{\mathcal{A}}(\mathcal{M})$, the *distance between the two states*, denoted $\Delta_{\mathcal{A},\mathcal{M}}(s_1, s_2)$ is defined iff there is an edge (s_1, s_2) in $G_{\mathcal{A}}(\mathcal{M})$ and its value is the weight of that edge.

2. A *path in $G_{\mathcal{A}}(\mathcal{M})$* is a sequence $s_1, s_2, \ldots, s_n$ such that for each i such that $1 \leq i \leq n$, $s_i \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$ and for each $j, 1 \leq j < n$, there is an edge (s_j, s_{j+1}) in $G_{\mathcal{A}}(\mathcal{M})$. The *length of a path* is the number of states in the path. The *cost of a path* is the sum of the weights of edges in the path. A *cycle in $G_{\mathcal{A}}(\mathcal{M})$* is a path $s_1, s_2, \ldots, s_n$ such that $s_1 = s_n$ and for all $1 \leq i \neq j < n, s_i \neq s_j$. The *path induced in $G_{\mathcal{A}}(\mathcal{M})$ by $\sigma(= r_1, \ldots, r_n)$ at* $s_0 \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$ is the sequence of states $s_0 s_1 \ldots s_n$ such that for all i, $0 \leq i < n$, there is an edge (s_i, s_{i+1}) labeled r_i in $G_{\mathcal{A}}(\mathcal{M})$. For any $s_0 \in G_{\mathcal{A}}(\mathcal{M})$, we say that σ *induces a cycle at* s_0 iff the path induced by σ in $G_{\mathcal{A}}(\mathcal{M})$ at s_0 is a cycle. (**Note**: $G_{\mathcal{A}}(\mathcal{M})$ could have paths and cycles with cost 0; see the note in Item 1 of this definition.)

3. A path $s_1, s_2, \ldots, s_n$ in $G_{\mathcal{A}}(\mathcal{M})$ is a *ρ-distinct path in $G_{\mathcal{A}}(\mathcal{M})$* iff there is a sequence of edges $(s_1, s_2), \ldots, (s_{n-1}, s_n)$ such that the cardinality of the set of labels of these edges is ρ and there is no sequence of edges $(s_1, s_2), (s_2, s_3) \ldots$ such that the cardinality of the set of labels of these edges is less than ρ. A cycle $s_1, s_2, \ldots, s_n (= s_1)$ in $G_{\mathcal{A}}(\mathcal{M})$ is a *ρ-distinct cycle in $G_{\mathcal{A}}(\mathcal{M})$* iff $s_1, s_2, \ldots, s_n (= s_1)$ is a ρ-distinct path.

The notes in Items 1 and 2 in the above definition point out that it is possible to have paths (cycles) in $G_{\mathcal{A}}(\mathcal{M})$ that have a cost of 0. Intuitively, as $\mathcal{A}$ changes states along a path with zero weight, $\mathcal{A}$ merely changes some of the bits in its memory (I), leaving the configurations of its servers unchanged. We will use this in Theorem 1 below.

We note that for a finite metric space $\mathcal{M}$, for each $I \geq 0$, and for each $k \geq 1$, $(\mathcal{C}^k_{\mathcal{M}} \times 2^I, D_{\mathcal{M}})$ is a symmetric metrical task system [2]. I-space bounded algorithms are possible candidate scheduling algorithms for this task system. Below, we *characterize* the set of I-space bounded algorithms that are competitive scheduling algorithms for this task system. Furthermore, we show that given an I-space bounded scheduling algorithm in a suitable form, the competitiveness of this algorithm in a given finite metric space is decidable!

Theorem 1 *Suppose $\mathcal{M}$ is a finite metric space, and $\mathcal{A}$ is a space bounded k-server algorithm. Then, $\mathcal{A}$ is competitive in $\mathcal{M}$ iff for all $\rho \leq k$, every ρ-distinct cycle in $G_{\mathcal{A}}(\mathcal{M})$ has cost 0.*

Proof. Suppose the hypothesis.

($\Longrightarrow$) Suppose $\mathcal{A}$ is competitive in $\mathcal{M}$. Now suppose by way of contradiction that $\rho \leq k$ and that there is a ρ-distinct cycle C in $G_{\mathcal{A}}(\mathcal{M})$ with cost greater than 0. Let $s_0 = (c_0, \iota_0)$ be a state in C. Let σ be a shortest request sequence which induces the cycle C in $G_{\mathcal{A}}(\mathcal{M})$ at s_0. Let $a_c = \mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s_0, \sigma)$. Clearly for every $i \geq 1$, $\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s_0, \sigma^i) = a_c \cdot i$. Let

$$R = \{r \mid \text{ there is an edge labeled } r \text{ in cycle } C\}.$$

Since C is a ρ-distinct cycle and $\rho \leq k$, it follows that the cardinality of R is at most k. Therefore, there exists c_f, a k-configuration, such that $R \subseteq c_f$. Clearly, $s = c_0 c_f c_f \ldots c_f \ldots$ is a k-service for σ^i starting at c_0. Also, $\mathtt{cost}_{\mathcal{M}}(s) = D_{\mathcal{M}}(c_0, c_f)$. It is also clear that s witnesses the fact that for all $i \geq 1$, $\mathtt{opt}^k_{\mathcal{M}}(c_0, \sigma^i) \leq D_{\mathcal{M}}(c_0, c_f)$. Therefore,

$$\lim_{i\to\infty} \frac{\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s_0, \sigma^i)}{\mathtt{opt}^k_{\mathcal{M}}(c_0, \sigma^i)} \geq \lim_{i\to\infty} \frac{a_c \cdot i}{D_{\mathcal{M}}(c_0, c_f)} = \infty.$$

Thus, $\mathcal{A}$ is not competitive, giving us a contradiction.

($\Longleftarrow$) Suppose every ρ-distinct cycle, $\rho \leq k$, in $G_{\mathcal{A}}(\mathcal{M})$ has cost 0. We will show that there exist two numbers, $\mathcal{B}$ and b, such that for all $s \in \mathcal{S}_{\mathcal{A}}(\mathcal{M}))$ and for all σ),

$$[\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) \leq \mathcal{B} \cdot \mathtt{opt}_{\mathcal{M}}(s, \sigma) + b].$$

Claim 1 *There exists b such that for every ρ-distinct sequence σ in $\mathcal{M}$, $\rho \leq k$ and for every $s \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$, $\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) \leq b$.*

Proof Claim 1. Suppose $\rho \leq k$ and σ is a ρ-distinct sequence in $\mathcal{M}$. Let $b = \sum_{s_i \neq s_j \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})} \Delta_{\mathcal{A},\mathcal{M}}(s_i, s_j)$. Suppose by way of contradiction that $s \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$ is such that $\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) > b$. Then, in the path in $G_{\mathcal{A}}(\mathcal{M})$ induced by σ at s, some edge of non-zero cost, say (s_1, s_2), is traversed at least twice (otherwise, from the definition of b, $\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma)$ cannot be greater than b). Thus, there is a σ' contained in σ such that σ' induces a cycle in $G_{\mathcal{A}}(\mathcal{M})$ at s_1. Let C be the cycle of shortest length containing edge (s_1, s_2) (Note that such a cycle must exist). Since σ is a ρ-distinct, it follows that C is a ρ'-distinct cycle, $\rho' \leq \rho \leq k$. Furthermore, the cost of this cycle is non-zero since $\Delta_{\mathcal{A},\mathcal{M}}(s_1, s_2) > 0$. Thus, we have a ρ'-distinct cycle, $\rho' \leq k$, of non-zero cost giving us a contradiction. $\square$ Claim 1

Let $\mathcal{D}$ be the diameter of $\mathcal{M}$. Let σ be any request sequence in $\mathcal{M}$, and let s be any state in $\mathcal{S}_{\mathcal{A}}(\mathcal{M})$.

Case 1: σ is a ρ-distinct sequence in $\mathcal{M}$, $\rho \leq k$. Then, by Claim 1, $\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) \leq b$.

Case 2: Let $\sigma = \sigma_0 \cdot \sigma_1 \cdot \ldots$, such that σ_0 is the empty sequence, and for every relevant $i \geq 1$, σ_i is the smallest $(k+1)$-distinct subsequence of σ following σ_{i-1}. (Note: In case σ is finite, or past a finite point is a ρ-distinct sequence in $\mathcal{M}$, $\rho \leq k$, then there will be only a finite number of values for i for which σ_i will be defined; hence we say "relevant i" above. In this case the trailing piece of σ, finite or infinite will be a ρ-distinct sequence, $\rho \leq k$.) Let $r_i \in \mathcal{M}$ such that $\sigma_i = \sigma'_i \cdot r_i$. Clearly, for every $i \geq 1$, $r_i \notin \sigma'_i$, and σ'_i is a k-distinct sequence. For each relevant $i \geq 1$, let $\sigma|_i$ denote the subsequence $\sigma_0 \cdot \sigma_1 \cdot \ldots \cdot \sigma_i$ of σ. Then, for every $i \geq 1$,

$$\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma|_i) \leq \mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma|_{i-1}) + k \cdot \mathcal{D} + b,$$

and, by Lemma 1

$$\mathtt{opt}_{\mathcal{M}}(s, \sigma|_i) \geq \mathtt{opt}_{\mathcal{M}}(s, \sigma|_{i-1}) + \mu.$$

Therefore, for every $i \geq 1$,

$$\frac{\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma|_i)}{\mathtt{opt}_{\mathcal{M}}(s, \sigma|_{i-1})} \leq \frac{k \cdot \mathcal{D} + b}{\mu}.$$

Furthermore, the trailing piece of σ (see the parenthetical note above), is a ρ-distinct sequence, $\rho \leq k$, and hence, by Claim 1, $\mathcal{A}$ will incur no more than a cost of b in servicing this trailing sequence. This sequence may not add to the optimal cost of servicing σ.

Therefore, combining Cases 1 and 2 above, we have that for every σ and every s in $\mathcal{S}_{\mathcal{A}}(\mathcal{M})$,

$$\mathtt{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) \leq \frac{k \cdot \mathcal{D} + b}{\mu} \mathtt{opt}_{\mathcal{M}}(s, \sigma) + b.$$

$\square$ THEOREM 1

We present two important consequences of this theorem below.

Theorem 2 *Suppose $\mathcal{A}$ is an arbitrary I-space bounded k-server algorithm on finite metric space $\mathcal{M}$. Then, it is decidable whether $\mathcal{A}$ is competitive on $\mathcal{M}$.*

PROOF. If $\mathcal{A}$ is an I-space bounded k-server algorithm on a finite metric space $\mathcal{M}$, then the graph $G_{\mathcal{A}}(\mathcal{M})$ induced by $\mathcal{A}$ on $\mathcal{M}$ is finite. Thus, from Theorem 1, verifying whether if $\mathcal{A}$ is competitive is equivalent to verifying that all ρ-distinct cycles, $\rho \leq k$, in $G_{\mathcal{A}}(\mathcal{M})$ have zero cost. Since the latter verification is effective, the theorem follows. $\square$

Consider the following algorithm for the k-server problem.

> If the request point r is currently occupied by any of the k servers, then no server moves; otherwise, service r using that server that was least recently moved to service a request in the past.

The above algorithm is essentially the *Least Recently Used* (*LRU*) algorithm which is used to implement memory paging in operating systems.

Theorem 3 *LRU is a competitive k-server algorithm for any finite metric space $\mathcal{M}$.*

PROOF.

It can be verified that all ρ-distinct cycles, $\rho \leq k$, in $G_{LRU}(\mathcal{M})$ have zero cost. Thus, from Theorem 1, it follows that *LRU* is competitive. $\square$

Acknowledgements

The authors would like to thank Errol Lloyd for extensive comments on an earlier draft of this paper, and Dayanand Rajan for many stimulating discussions and suggestions.

References

[1] Ganesh R. Baliga and Anil M. Shende. Memoryless 2-server algorithms are not 2-competitive. Technical Report 92-6, Dept. of Computer Science, Bucknell University, July 1992.

[2] Allan Borodin, Nathan Linial, and Michael Saks. An optimal online algorithm for metrical task systems. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 373–382, 1987.

[3] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291–300, 1990.

[4] Marek Chrobak and Lawrence L. Larmore. A new approach to the server problem. *SIAM Journal on Discrete Mathematics*, 4(3):323–328, August 1991.

[5] Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM Journal on Computing*, 20(1):144–148, February 1991.

[6] Marek Chrobak and Lawrence L. Larmore. Generosity helps, or an 11-competitive algorithm for three servers. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 196–202, 1992.

[7] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for on-line problems. In *STOC*, pages 322–333, 1988.