

# On Space Bounded Server Algorithms

Ganesh R. Baliga  
(email: baliga@rowan.edu)  
Department of Computer Science  
Rowan College of New Jersey  
Glassboro, NJ 08028  
USA

Steven Hughes  
(email: sthughes@cs.indiana.edu)  
Department of Computer Science  
Indiana University  
Bloomington, IN  
USA

Anil M. Shende\*  
(email: shende@bucknell.edu)  
Department of Mathematics, Computer Science & Physics  
Roanoke College  
Salem, VA 24153  
USA

---

\*This work was partially supported by a Mellon grant at Dickinson College during the Summer 1994.

## Abstract

This paper studies *space bounded* algorithms, i.e., algorithms that use a constant amount of scratch space, for the  $k$ -server problem. The paper first shows a characterization of the set of space bounded algorithms that are competitive in finite metric spaces. The result yields a method for (1) effectively deciding if a given space bounded algorithm is competitive in a given finite metric space, and (2) for designing competitive space bounded algorithms in finite metric spaces.

The above results are then refined, in particular using the geometry of metric spaces, to develop two methods for obtaining an upper bound on competitive factors of arbitrary space bounded algorithms. One of these methods is then used to show that any algorithm, employing the Least Recently Used strategy, is linearly competitive on a uniform metric space.

## 1 Introduction

On-line problems arise quite often, and quite naturally in computer science. The dynamic nature of these problems demands algorithms for these problems to make decisions without full knowledge of the impact of their decisions on the future performance of these algorithms. Many data structuring problems and scheduling problems are on-line problems. [5] proposed an abstract model for capturing on-line problems, and subsequently [18] formulated a variant of on-line problems—the  $k$ -server problem.  $k$ -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. Essentially, the  $k$ -server problem is to come up with a  $k$ -server algorithm which, given as input a point, (typically called a *request*), in a metric space, moves at least one of its  $k$  servers to that point (typically called *servicing the request*), and then waits for the next request. Furthermore, the amount this algorithm moves its servers, also called the *cost* of the algorithm, should be as little as possible. [18] conjectured that for each  $k$ , there is a  $k$ -server algorithm that, for any request sequence, does not have a cost more than  $k$  times the optimal cost necessary to service the requests—this is now known as the  $k$ -server conjecture.  $k$ -server algorithms that have a cost within a constant factor of the optimal cost are called *competitive*, and the factor is called the *competitive factor* of the algorithm.

For completeness, we first formally define the  $k$ -server problem and some related notions in Section 2. In Section 2.2 we present a literature survey of the work that has been done on this problem. As will be seen there, the  $k$ -server conjecture remains unresolved.

Our work, presented in Sections 3, is, in a lot of ways, along the lines of [5] and [18] mentioned above. In Section 3 we focus on  $k$ -server algorithms, that use a constant amount of scratch space, for the  $k$ -server problem (we call these *space bounded* algorithms) in finite metric spaces. Most real-life problems that can be modeled as  $k$ -server problems get requests from a finite metric space. Also, an algorithm that takes up arbitrarily large amounts of scratch space, e.g., algorithms that take into account their decisions on *all* the past requests before making a decision for the current request, are unlikely to make their decisions in real-time, thus making them impractical for on-line problems. [5] gives an upper bound on the competitive factor

of any space bounded algorithm in a finite metric space; [18] presents algorithms for 2 and  $(n - 1)$  servers, leaving open the search for competitive space bounded  $k$ -server algorithms ( $2 < k < n - 1$ ) in finite metric spaces with  $n$  points. We essentially settle this question by characterizing 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 competitive space bounded algorithms for any given space bound and any finite metric space!

In Section 4, we provide two methods to compute upper bounds on the competitive factors of competitive space-bounded algorithms operating on finite metric spaces. In particular, focusing on the geometry of metric spaces, we show that the competitive factor of the LRU paging strategy is linear in the number of servers (pages).

## 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  $r_1, r_2, \dots, r_i, \dots$ . 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  $\sigma$  and  $\sigma'$  to denote *request sequences* which are essentially strings in  $\mathcal{M}^*$ . For all  $\sigma$  and finite  $\sigma'$ ,  $\sigma' \cdot \sigma$  will denote the request sequence comprising first of the requests in  $\sigma'$  followed by requests in  $\sigma$ . For a sequence of requests  $\sigma = r_1 r_2 \dots r_n$  in  $\mathcal{M}$ , for each  $i \geq 1$ ,  $\sigma^i$  denotes the request sequence  $\sigma$  repeated  $i$  times.

### 2.1 Definitions and Notation

#### Definition 1

1. A  $k$ -configuration in  $\mathcal{M}$  is a multiset of cardinality  $k$  and comprising of points in  $\mathcal{M}$ .  $\mathcal{C}_{\mathcal{M}}^k$  denotes the set of all  $k$ -configurations in  $\mathcal{M}$ .
2. A *matching*  $f_{c_1, c_2}$  between  $k$ -configurations  $c_1, c_2 \in \mathcal{C}_{\mathcal{M}}^k$  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  $c_1, c_2 \in \mathcal{C}_{\mathcal{M}}^k$ ,  $D_{\mathcal{M}}(c_1, c_2)$  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  $c_0 c_1 c_2 \dots$ , such that for all  $i \geq 1, c_i \in \mathcal{C}_{\mathcal{M}}^k$  is a  $k$ -service in  $\mathcal{M}$ . We will also say that this  $k$ -service in  $\mathcal{M}$  starts at  $c_0$ .
4. The *cost of a  $k$ -service*  $s = c_0 c_1 c_2 \dots$  in  $\mathcal{M}$ , denoted  $\text{cost}_{\mathcal{M}}(s)$ , is given by

$$\text{cost}_{\mathcal{M}}(s) = D_{\mathcal{M}}(c_0, c_1) + D_{\mathcal{M}}(c_1, c_2) + \dots$$

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

6. For all  $c \in \mathcal{C}_{\mathcal{M}}^k$ , for all  $\sigma$ , the *optimal cost in  $\mathcal{M}$  for servicing  $\sigma$  starting at  $c$  with  $k$  servers*, denoted  $\text{opt}_{\mathcal{M}}^k(c, \sigma)$ , is the minimum cost over all  $k$ -services in  $s_{\mathcal{M}}^k(c, \sigma)$ .

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

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

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  $c \in \mathcal{C}_{\mathcal{M}}^k$  and for all  $\sigma$ ,

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

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  $\mathcal{B}$ . A  $k$ -server algorithm is  $\mathcal{B}$ -competitive (competitive) iff it is  $\mathcal{B}$ -competitive (competitive) in *all* metric spaces  $\mathcal{M}$ .

We now isolate a property of optimal servicing schedules for request sequences in a large class of metric spaces. The metric spaces  $\mathcal{M}$  considered herein are those for which there exists a  $\mu > 0$  such that

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

Finite metric spaces clearly constitute an example of such metric spaces.

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

**Lemma 1** Suppose  $\mathcal{M}$  is a metric space wherein the distance between any two distinct points is at least  $\mu > 0$ . For each  $k \geq 1$ , let  $\Sigma_k$  be the set of all  $(k + 1)$ -distinct sequences in  $\mathcal{M}$ . Then, for every  $k \geq 1$ , for every  $c \in \mathcal{C}_{\mathcal{M}}^k$ , for every finite  $\sigma$  and every  $\sigma' \in \Sigma_k$ ,  $\text{opt}_{\mathcal{M}}^k(c, \sigma \cdot \sigma') \geq \text{opt}_{\mathcal{M}}^k(c, \sigma) + \mu$ .

PROOF. Fix  $c, k, \sigma$  and  $\sigma' = r_1 r_2 \dots$  as above. Let  $s_{\sigma, \sigma'}$  be an arbitrary  $k$ -service for  $\sigma$  starting at  $c$ . Then,  $s_{\sigma, \sigma'}$  can be written as  $c \cdot s_{\sigma} \cdot s_{\sigma'}$  where  $c \cdot s_{\sigma}$  is a  $k$ -service for  $\sigma$  starting at  $c$ , and,  $c_{\sigma} \cdot s_{\sigma'} = c_{r_1} c_{r_2} \dots$  is a  $k$ -service for  $\sigma'$ , where  $c_{\sigma}$  is the last configuration in  $s_{\sigma}$ . Note first that  $\text{cost}_{\mathcal{M}}(c \cdot s_{\sigma}) \geq \text{opt}_{\mathcal{M}}^k(c, \sigma)$ . Let  $r_i$  be the first

request point in  $\sigma'$  such that  $r_i \notin c_\sigma$ . Since  $\sigma'$  is a  $(k + 1)$ -distinct sequence and since  $c_\sigma \in \mathcal{C}_{\mathcal{M}}^k$ , from the pigeonhole principle we can conclude that such an  $r_i$  exists. It now follows that either  $D_{\mathcal{M}}(c_\sigma, c_{r_1}) \neq 0$  or there exists  $j$ ,  $1 \leq j < i$  such that  $D_{\mathcal{M}}(c_{r_j}, c_{r_{j+1}}) \neq 0$ . Hence, it follows that  $\text{cost}_{\mathcal{M}}(c_\sigma \cdot \sigma') \geq \mu$ , since any matching of  $k$ -configurations in  $\mathcal{M}$  with non-zero weight has weight at least  $\mu$ .

Thus  $\text{opt}_{\mathcal{M}}^k(c, \sigma \cdot \sigma') \geq \text{cost}_{\mathcal{M}}(c \cdot s_{\sigma, \sigma'}) = \text{cost}_{\mathcal{M}}(c \cdot s_\sigma) + \text{cost}_{\mathcal{M}}(c_\sigma \cdot s_{\sigma'}) \geq \text{opt}_{\mathcal{M}}^k(c, \sigma) + \mu$ .  $\square$

## 2.2 Survey of Results on the $k$ -Server Problem

The  $k$ -server problem was proposed in [18] as a variant of the *metrical task systems* studied in [5]. Metrical task systems are essentially domains of dynamic systems; the systems themselves, in the terminology of [5], are *scheduling algorithms*. A metrical task system is a pair  $(S, d)$ , where  $S$  is a collection of states, and the state transition cost function  $d$  is a function that associates a real number with every ordered pair of states. A metrical task system is said to be *symmetric* iff  $d$  is symmetric. As a scheduling algorithm is given tasks, it changes from one state in  $S$  to another; with each such change of state there is an associated cost (given by  $d$ ). The scheduling algorithm makes its decision of which state to go to based on its current state and the task at hand; clearly, these decisions are made without full knowledge of their future impact on the costs to be incurred by the algorithm.

Prior to this formulation of metrical task systems, [19] studied an *on-line* algorithm (essentially a scheduling algorithm), “move-to-front”, for dynamically maintaining a linear list, and showed that the amortized performance (proposed in [20]) of this algorithm is within a constant factor of the optimal for this problem, i.e., the best performance over *all* algorithms, including *off-line* algorithms, for this problem. [15] coined the term *competitive* for such algorithms, i.e., those on-line algorithms, for a given problem, whose amortized performance is within a constant factor (*competitive factor*) of optimal cost (off-line cost) for that problem, and devised competitive caching strategies for snoopy caching systems.

[5] adopted the notion of competitive algorithms, and showed that in general for every metrical task system with  $n$  states, there is a competitive scheduling algorithm with competitive factor no more than  $O(n^2)$ , and in particular, for symmetric metrical task systems, this factor is no more than  $2n - 1$ . (As will be pointed out later, our model for finite metric spaces and space bounded server algorithms is essentially a symmetric metrical task system. Thus these results apply to our model too.)

Most practical problems such as planning the motion of the several heads in a multi-headed disk under a sequence of requests, and the paging problem do not need the generality of metrical task systems, and can be formulated as server problems [18]. As far as competitive on-line server algorithms are concerned, [18] showed the serious limitation that the competitive factor of a  $k$ -server algorithm can be no lesser than  $k$  if the metric space of requests has at least  $k + 1$  points! They also conjectured, what is now well-known as the  *$k$ -server conjecture*, that for each  $k$ , there is a  $k$ -competitive (competitive factor of  $k$ )  $k$ -server on-line algorithm. The truth of this conjecture still remains an open question. [18] also devised a 2-server algorithm, and showed,

using what they call the method of *residues*, that this algorithm is 2-competitive for symmetric metric spaces. In [18] there is also a study of the server problem where the servers are allowed *excursions* and the metric space of requests is finite; they resolve the cases of 2 and  $(n-1)$  servers in a metric space with  $n$  points, and mention analogous results for less than  $(n-1)$  servers as an open problem. Our work, presented in Section 3 below, sheds some light on this problem.

For practical problems, it is imperative that on-line algorithms make their decisions in real-time. It thus seems natural to study algorithms that do not take into account a lot of past history or possible future requests. *Memoryless* on-line algorithms, a special case of such possibly fast on-line algorithms, that do not base their decisions on anything other than the current request to be serviced (in the terminology of the  $k$ -server problem) and the current positions of the servers, were studied in [6]. They presented a  $k$ -server algorithm DOUBLE COVERAGE for requests on the real line  $\mathbf{R}^1$ , and showed, using amortized analysis, that this algorithm is  $k$ -competitive; furthermore, they isolated a necessary condition for memoryless algorithms to be competitive on the real line. In [9], it was proved that an analog of DOUBLE COVERAGE is  $k$ -competitive on trees (with path length as the associated metric). [6] also presented a  $k$ -competitive  $k$ -server algorithm for asymmetric metric spaces—the *weighted cache problem*, and showed that there is no memoryless competitive algorithm for this problem.

[7] proposed a new approach for  $k$ -server algorithms. Essentially, the algorithm moves its servers in a virtual metric space, keeping track of all relevant history in the positions of the servers in the virtual metric space. They first showed [7] that their 2 server algorithm which uses this approach is 2-competitive on any metric space; they also showed [10] that this approach can be generalized to obtain a 11-competitive algorithm for the 3-server problem. As yet no  $k$ -competitive algorithms for  $k$  servers,  $k > 2$  are known that work on all metric spaces; recently, the existence of a  $(2k - 1)$ -competitive algorithm has been shown [17].

Towards fast, on-line algorithms, [8] presented a 4-competitive algorithm for 2 servers on any metric space. Most often, in practice, problems requiring on-line algorithms have finite metric spaces, e.g., the paging problem. Early on, [18] studied on-line algorithms on finite metric spaces, and presented an  $(n-1)$ -competitive  $(n-1)$ -server algorithm for a metric space with  $n$  points. They also presented a 2-competitive 2-server algorithm for finite metric spaces. The results in [5] had already established the existence of competitive algorithms on finite metric spaces; [18] presents explicit competitive algorithms for 2 servers and  $(n-1)$  servers. In Section 3 below, we present a characterization for the competitiveness of a class of possibly fast on-line  $k$ -server algorithms. We call these algorithms *space bounded*. Space bounded algorithms service requests based on the current configuration of servers, the contents of its finite *bounded* memory and the position of the current request. Furthermore, our characterization yields an algorithm for deciding whether a given space bounded  $k$ -server algorithm is competitive on a given finite metric space.

The server problem has been used as a model to study algorithms for various other problems. For example, [4] motivated by applications in data compression, debugging and physical simulation, studied on-line algorithms for locating checkpoints

in long computations, by formulating this as a server problem. [14] studied the specific problem of paging and proved that the competitiveness of certain algorithms to implement paging improves if the programs being executed (and hence causing page faults, etc.) are structured and hence exhibit locality of reference.

In recent years, there has been a growing interest in devising *randomized k-server* algorithms. Although randomized algorithms are not relevant to this paper, we summarize some of this work here. [3] showed that the power of randomization in on-line algorithms is essentially limited by proving the existence of an efficient simulation of randomized on-line algorithms by deterministic ones! [11] presented a randomized  $k$ -competitive  $k$ -server algorithm on a class of metric spaces they call *resistive*. [16] present a randomized algorithm for the bipartite matching problem, and show that it achieves the best possible performance among on-line algorithms for the problem. [1] studies a variant of this problem, and shows that the competitive factor for the best possible deterministic on-line algorithm differs from the competitive factor for the best possible randomized algorithm for this problem by only a constant.

### 3 Space Bounded $k$ -Server Algorithms

#### Definition 4

1. We say that an algorithm  $\mathcal{A}$  is  *$I$ -space bounded* iff on any input,  $\mathcal{A}$  uses at most  $I$  bits of memory in its computation.
2. We say that algorithm  $\mathcal{A}$  is *space bounded* iff it is  $I$ -space bounded for some  $I$ .

In this section, we study space bounded  $k$ -server algorithms on finite metric spaces  $\mathcal{M}$ . Examples of space bounded algorithms are algorithms which perform bounded look back on the request sequence. Competitive space bounded  $k$ -server algorithms are desirable, since, in practical applications of the  $k$ -server problem such as multi-head disk scheduling, request sequences can grow arbitrarily long in a short time span and hence algorithms which are not bounded in their space usage would certainly fail to provide real time solutions!

Let  $\mathcal{A}$  be any arbitrary  $I$ -space bounded  $k$ -server algorithm on  $\mathcal{M}$ . Prior to servicing request  $R$ ,  $\mathcal{A}$ '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 which  $\mathcal{A}$  uses. Thus  $\mathcal{A}$  can in fact be in one of  $\mathcal{C}_{\mathcal{M}}^k \times 2^I$  states.  $\mathcal{A}$  will service  $R$  based on its current state and the position of request  $R$  in  $\mathcal{M}$ . We now proceed formally.

#### 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  $(\mathcal{C}_{\mathcal{M}}^k \times 2^I)$ . 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  $r \in M$  such that  $f_{\mathcal{A}}((c_1, \iota_1), r) = (c_2, \iota_2)$ . Furthermore, this edge is assigned a weight equal to  $D_{\mathcal{M}}(c_1, c_2)$ . Notice that all edges between two vertices have the same

weight. 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 state  $s$  in  $\mathcal{S}_{\mathcal{A}}(\mathcal{M})$ ,  $C(s)$  denotes the configuration of the servers in  $s$ . 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 sequence  $s_1, s_2, \dots, s_i \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$ , such that for each  $i \geq 1$ ,  $s_i \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$  and there is an edge  $(s_i, s_{i+1})$  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 finite path  $s_1, s_2, \dots, s_n$  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  $\sigma$  at  $s_0 \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$* , denoted by  $P_{\mathcal{A}}(\mathcal{M}, s_0, \sigma)$ , is the sequence of states  $s_0 s_1 \dots s_n$  such that for all  $i, 0 \leq i < n$ , there is an edge  $(s_i, s_{i+1})$  labeled  $r_i$  in  $G_{\mathcal{A}}(\mathcal{M})$ .  $\sigma$  *induces a cycle at  $s_0 \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$*  iff the path induced by  $\sigma$  in  $G_{\mathcal{A}}(\mathcal{M})$  at  $s_0$  is a cycle.
3. A path  $s_1, s_2, s_3 \dots$  in  $G_{\mathcal{A}}(\mathcal{M})$  is a  $\rho$ -*distinct path in  $G_{\mathcal{A}}(\mathcal{M})$*  iff there is a sequence of edges  $(s_1, s_2), (s_2, s_3), \dots$  such that the cardinality of the set of labels of these edges is  $\rho$  and there is no sequence of edges  $(s_1, s_2), (s_2, s_3) \dots$  such that the cardinality of the set of labels of these edges is less than  $\rho$ . A cycle  $s_1, s_2, \dots, s_n (= s_1)$  in  $G_{\mathcal{A}}(\mathcal{M})$  is a  $\rho$ -*distinct cycle in  $G_{\mathcal{A}}(\mathcal{M})$*  iff  $s_1, s_2, \dots, s_n (= s_1)$  is a  $\rho$ -distinct path.

We note here that for a finite metric space  $\mathcal{M}$ , for each  $I \geq 0$ , and for each  $k \geq 1$ ,  $(\mathcal{C}_{\mathcal{M}}^k \times 2^I, D_{\mathcal{M}})$  is a symmetric metrical task system [5].  $I$ -space bounded algorithms are possible candidate scheduling algorithms for this task system. As noted earlier in Section 2.2, [5] asserts that for each such task system, there is an  $I$ -space bounded algorithm that is competitive with a competitive factor no more than  $(2|\mathcal{M}| - 1)$ . 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  $\rho$ -distinct cycle in  $G_{\mathcal{A}}(\mathcal{M})$  has cost 0.*

PROOF. Suppose the hypothesis.

( $\implies$ ) Suppose  $\mathcal{A}$  is competitive in  $\mathcal{M}$ . Now suppose by way of contradiction that  $\rho \leq k$  and that there is a  $\rho$ -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  $\sigma$  be the shortest request sequence which induces cycle  $C$  in  $G_{\mathcal{A}}(\mathcal{M})$  at  $s_0$ . Let  $a_c = \text{COST}_{\mathcal{M}}(\mathcal{A}, s_0, r_0 r_1 \dots r_n)$ . Clearly for every  $i \geq 1$ ,  $\text{COST}_{\mathcal{M}}(\mathcal{A}, s_0, \sigma) = a_c \cdot i$ . Let

$$R = \{r \mid (\exists s_i, s_j \in G_{\mathcal{A}}(\mathcal{M}))[(s_i, s_j) \text{ is an edge, labeled } r, \text{ in cycle } C]\}.$$

Since  $C$  is a  $\rho$ -distinct cycle,  $\rho \leq k$ , it follows that the cardinality of  $R$  is at most  $k$ . Let  $c_f$  be a  $k$ -configuration such that  $R \subseteq c_f$ . Now consider the  $k$ -service  $s = c_0 c_f c_f \dots c_f \dots$ . Clearly  $\text{cost}_{\mathcal{M}}(s) = D_{\mathcal{M}}(c_0, c_f)$ . It is also clear that  $s$  witnesses the fact that for all  $i \geq 1$ ,  $\text{opt}_{\mathcal{M}}^k(c_0, \sigma^i) \leq D_{\mathcal{M}}(c_0, c_f)$ . Therefore,

$$\lim_{i \rightarrow \infty} \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s_0, \sigma^i)}{\text{opt}_{\mathcal{M}}^k(c_0, \sigma^i)} = \lim_{i \rightarrow \infty} \frac{a_c \cdot i}{D_{\mathcal{M}}(c_0, c_f)} = \infty.$$

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

( $\Leftarrow$ ) Suppose every  $\rho$ -distinct cycle,  $\rho \leq k$ , in  $G_{\mathcal{A}}(\mathcal{M})$  has cost 0. We will show that there exists a  $\mathcal{B}$  and  $b$  such that

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

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

PROOF CLAIM 1. 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  $\rho \leq k$  and  $\sigma$  is a  $\rho$ -distinct sequence in  $\mathcal{M}$ . Suppose by way of contradiction that  $s \in \mathcal{S}_{\mathcal{A}}(\mathcal{M})$  is such that  $\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) > b$ . Then, in the path in  $G_{\mathcal{A}}(\mathcal{M})$  induced by  $\sigma$  at  $s$ , some edge of non-zero cost, say  $(s_1, s_2)$ , is traversed at least twice (otherwise, from the definition of  $b$ ,  $\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma)$  cannot be greater than  $b$ ). Thus, there is a  $\sigma'$  contained in  $\sigma$  such that  $\sigma'$  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  $\sigma$  is a  $\rho$ -distinct, it follows that  $C$  is a  $\rho'$ -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  $\rho'$ -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  $\sigma$  be any request sequence in  $\mathcal{M}$ , and let  $s$  be any state in  $\mathcal{S}_{\mathcal{A}}(\mathcal{M})$ .

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

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

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

and, by Lemma 1

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

Therefore, for every  $i \geq 1$ ,

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

Furthermore, the trailing piece of  $\sigma$  (see the parenthetical note above), is a  $\rho$ -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  $\sigma$ .

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

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

□ 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  $\rho$ -distinct cycles,  $\rho \leq k$ , in  $G_{\mathcal{A}}(\mathcal{M})$  have zero cost. Since the latter verification is effective, the theorem follows. □

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 is easy to verify that all  $\rho$ -distinct cycles,  $\rho \leq k$ , in  $G_{LRU}(\mathcal{M})$  have zero cost. Thus, from Theorem 1, it follows that *LRU* is competitive. □

## 4 Upper Bounds for the Competitive Factor

Competitive factors have been proven for specific algorithms with specific metric spaces. Intuitively, calculating the competitive factor of an algorithm involves the examination of an infinite number of possible request sequences. Clearly, this is intractable. This section introduces two finite methods for calculating upper bounds on the competitive factor of arbitrary space bounded algorithms.

For the rest of this paper, let  $\mathcal{M}$  be an arbitrary finite metric space with  $m$  points, and let  $\mathcal{A}$  be a space bounded (no more than  $b$  bits)  $k$ -server algorithm that is competitive in  $\mathcal{M}$ .  $\sigma$  will denote request sequences consisting of points in  $\mathcal{M}$  and  $s$  will denote states in  $S_{\mathcal{A}}(\mathcal{M})$ . Recall the definitions and notation from Sections 2.1 and 3. We also adopt standard graph theoretic definitions of *closed walks* (also called *circuits* in the literature).

**Definition 6** A *closed walk* in  $G_{\mathcal{A}}(\mathcal{M})$  is a sequence of states  $s_1, \dots, s_t$  in  $S_{\mathcal{A}}(\mathcal{M})$  such that for each  $i$ ,  $1 \leq i < t$ ,  $(s_i, s_{i+1})$  is an edge in  $G_{\mathcal{A}}(\mathcal{M})$ , and  $s_1 = s_t$ .

Note that a cycle (see Definition 5) is just a closed walk so that *only* the first and last states in the walk are the same.

### 4.1 Upper Bounds Based on Cycles

**Lemma 2** For all  $s$  in  $S_{\mathcal{A}}(\mathcal{M})$  and all  $\sigma$ ,  $P_{\mathcal{A}}(\mathcal{M}, s, \sigma)$  can be divided into a finite number of closed walks and remaining states that do not contain a cycle.

PROOF. Any repeated states in  $P_{\mathcal{A}}(\mathcal{M}, s, \sigma)$  represent a closed walk. The first state repeated identifies a cycle. Remove cycles from  $P_{\mathcal{A}}(\mathcal{M}, s, \sigma)$ , retaining the initial point. Now parse through the remainder states, again removing cycles. Repeat the process until no cycles are found. With each iteration the size of  $P_{\mathcal{A}}(\mathcal{M}, s, \sigma)$  is decreasing, thus we know this is a finite process. Inductively, at the end of each iteration, we can see that  $P_{\mathcal{A}}(\mathcal{M}, s, \sigma)$  is composed of a finite number of closed walks and some remaining states. When the process stops, the remaining states have no cycles and hence the lemma.  $\square$

For any closed walk  $y$  in  $G_{\mathcal{A}}(\mathcal{M})$  and any  $s$  in  $y$ , let  $\sigma_{y,s}$  be a request sequence that induces  $y$ , starting at  $s$ . Note that  $\forall s, s'$  in  $y$ ,  $\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_{y,s}) = \text{COST}_{\mathcal{M}}(\mathcal{A}, s', \sigma_{y,s'})$ . Therefore, when referring to closed walks, we can refer to  $\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_{y,s})$  simply as  $\text{COST}_{\mathcal{M}}(\mathcal{A}, y)$ .

The cost incurred by an optimal servicing of a request sequence can vary depending on it's initial state. However, over all possible states, there is a minimum cost for servicing any given request sequence. Let  $\mu(\mathcal{M}, \sigma)$  denote this minimum cost of servicing  $\sigma$ .

In particular for a closed walk,  $y$ , every state  $s$  in  $y$  has a corresponding request sequence,  $\sigma_{y,s}$ , that induces  $y$  in  $G_{\mathcal{A}}(\mathcal{M})$ . Abusing notation, let

$$\text{COST}_{\mathcal{M}}(\mathcal{A}, y) = \min\{\mu(\mathcal{M}, \sigma_{y,s}) \mid s \in y\}.$$

Since every cycle is a closed walk, the above notation applies to cycles as well. For a cycle  $s$ , let  $R_s = \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s)}{\text{opt}_{\mathcal{M}}(s)}$ . Then, let  $R = \max\{R_s\}$ .

**Theorem 4** *The competitive factor of  $\mathcal{A}$  operating in  $\mathcal{M}$  is bounded above by  $R$ .*

PROOF. Let  $w$  denote the sum of the weights of all arcs in  $G_{\mathcal{A}}(\mathcal{M})$ . In this proof,  $s$  will range over cycles, and  $y$  will range over closed walks.

By definition of  $R$ ,  $\forall s \in G_{\mathcal{A}}(\mathcal{M})$ ,  $\text{COST}_{\mathcal{M}}(\mathcal{A}, s) \leq R \cdot \text{opt}_{\mathcal{M}}(s)$ .

Furthermore, for the composition of any  $v$  cycles,  $s_1, s_2, \dots, s_v$  in  $G_{\mathcal{A}}(\mathcal{M})$ , we have

$$\sum_{i=1}^v \text{COST}_{\mathcal{M}}(\mathcal{A}, s_i) \leq R \cdot \left( \sum_{i=1}^v \text{opt}_{\mathcal{M}}(s_i) \right).$$

However, a closed walk is the composition of cycles. Therefore,  $\text{COST}_{\mathcal{M}}(\mathcal{A}, y) \leq R \cdot \text{opt}_{\mathcal{M}}(y)$ . For all  $\sigma$ , by Lemma 2, it is known that  $P_{\mathcal{A}}(\mathcal{M}, c, \sigma)$  is made up of a finite number of closed walks,  $y_1, y_2, \dots, y_t$ , plus a remainder path. Let  $\sigma_v$  be the part of  $\sigma$  inducing the remainder path. Then,

$$\begin{aligned} \text{COST}_{\mathcal{M}}(\mathcal{A}, c, \sigma) &\leq \sum_{i=1}^t \text{COST}_{\mathcal{M}}(\mathcal{A}, y_i) + \text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_v), \text{ and} \\ \text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma) &\geq \sum_{i=1}^t \text{opt}_{\mathcal{M}}(y_i). \end{aligned}$$

Therefore,

$$\begin{aligned} \text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) &\leq \sum_{i=1}^t \text{COST}_{\mathcal{M}}(\mathcal{A}, y_i) + \text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_v) \\ &\leq R \cdot \sum_{i=1}^t \text{opt}_{\mathcal{M}}(y_i) + \text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_v) \\ &\leq R \cdot \text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma) + \text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_v) \\ &\leq R \cdot \text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma) + w. \end{aligned}$$

Then, by definition, the competitive factor of the algorithm is no more than  $R$ .  $\square$

By Theorem 4, we now know that to calculate an upper bound for the competitive factor for a space bounded algorithm, we only need to look at the  $\rho$ -cycles, that it can generate, where  $\rho > k$ .

## 4.2 Upper Bounds Based on the Geometry of the Metric Space

While Theorem 4 will give us a very good upper bound, calculating it is still a formidable task. As we examine larger metric spaces, the number of candidate closed walks grows exponentially. This section provides another upper bound. Although not

as accurate, it exploits both the geometry of the metric space, as well as the properties of the algorithm. Hence, it yields good upper bounds on the competitive factor of specific algorithms on specific metric spaces.

**Definition 7**

1. For every pair of states,  $s_1, s_2$ , let  $\psi(s_1, s_2)$  be the length of the longest request sequence  $\sigma$  such that (1)  $\sigma$  consists solely of request points in  $C(s_2)$ , and (2)  $\mathcal{A}(s_1, \sigma) \neq s_2$  and there exists a request point  $r$  such that  $\mathcal{A}(s_1, \sigma \cdot r) = s_2$ . Then, the *convergence factor* of  $\mathcal{A}$ , denoted  $\Psi$  is defined as

$$\Psi = \max\{\psi(s_1, s_2) \mid s_1, s_2 \in G_{\mathcal{A}}(\mathcal{M})\}.$$

2.  $\sigma$  is a *short* request sequence iff there exists a configuration,  $g$  such that starting at  $g$ , the adversary can service  $\sigma$  by moving exactly one server once.  $\psi_1$  denotes the set of all short request sequences.
- 3.

$$R_m(\sigma) = \max \left\{ \frac{C_P(c, \sigma)}{C_A(g, \sigma)} \mid c \in G(M, P), \right. \\ \left. g \text{ witnesses } \sigma \text{ is short} \right\}$$

The following lemma follows from the definition of  $\Psi$  and  $\psi_1$ , and justifies the definition of  $R_m$  below (see Definition 8).

**Lemma 3** *For all  $\sigma \in \psi_1$ , such that  $|\sigma| > 2\Psi$ , there is a  $\sigma' \in \psi_1$  such that  $|\sigma'| \leq 2\Psi$  and  $R_m(\sigma) = R_m(\sigma')$ .*

PROOF. Let  $\sigma \in \psi_1$  (as witnessed by configuration  $g$ ) such that  $|\sigma| > 2\Psi$ . After servicing  $\sigma$ , starting at configuration  $g$ , the adversary must be in a configuration that differs from  $g$  in exactly one point. Let  $\sigma = \sigma_1 \cdot \sigma_2$  such that the adversary does not move any server for servicing  $\sigma_1$ , moves one server to service the first request in  $\sigma_2$ , and does not move any server to service the rest of  $\sigma_2$ . Clearly, each of  $\sigma_1$  and  $\sigma_2$  consist of only  $k$  distinct points. Since  $|\sigma| > 2\Psi$ , at least one of  $\sigma_1$  and  $\sigma_2$  must have cardinality more than  $\Psi$ . From the definition of  $\Psi$  it follows that if a request sequence,  $\sigma$ , consists of only  $k$  distinct points and the player, starting at some state  $c$ , incurs a non-zero cost for servicing each of the requests in  $\sigma$ , then  $|\sigma| \leq \Psi$ . This implies that the cost incurred by the player, starting at state  $c$ , to service the request sequence  $\sigma' = \sigma'_1 \cdot \sigma'_2$  where  $\sigma'_1$  is the first  $\Psi$  requests in  $\sigma_1$  (all of  $\sigma_1$ , if it has less than  $\Psi$  requests) and  $\sigma'_2$  is the first  $\Psi$  requests in  $\sigma_2$  (all of  $\sigma_2$ , if it has less than  $\Psi$  requests) is the same as for servicing  $\sigma$ . Thus  $R_m(\sigma') = R_m(\sigma)$ .  $\square$

**Definition 8**  $R_m = \max\{R_m(\sigma) \mid \sigma \in \psi_1\}$ .

The following lemma shows that a bound on the competitive factor of a space bounded algorithm can be computed simply by considering all the request sequences in  $\psi_1$  defined above. Furthermore, Lemma 3 above implies that the number of such request sequences that need to be considered is finite.

**Lemma 4** *The competitive factor for  $\mathcal{A}$  operating in  $\mathcal{M}$  is bounded above by  $R_m$ .*

PROOF. For all  $\sigma$  in  $\psi_1$ , the cost of the algorithm for servicing  $\sigma$  is clearly bounded above by  $R_m$  times the cost of the optimal for servicing  $\sigma$ . Suppose  $\sigma \notin \psi_1$ . Then,  $\sigma$  can be broken down into subsequences  $\sigma_1, \sigma_2, \dots, \sigma_n$  such that  $(\forall i \mid 1 \leq i \leq n)[\sigma_i \in \psi_1]$ . Therefore,

$$\frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma)}{\text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma)} = \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_1 \cdot \sigma_2 \cdot \dots \cdot \sigma_n)}{\text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma_1 \cdot \sigma_2 \cdot \dots \cdot \sigma_n)}$$

Let  $s$  be the starting state for  $\mathcal{A}$  and for each  $i$ ,  $1 \leq i < n$ , let  $s_{i+1} = \mathcal{A}(s_i, \sigma_i)$ . Let  $g_1 = \mathcal{C}(s)$ , and for each  $i$ ,  $1 < i \leq n$  let  $g_i$  denote the configuration of the adversary just before servicing  $\sigma_i$ . Let  $t$  be such that,

$$(\forall i \mid 1 \leq i \leq n) \left[ \frac{C_P(s_t, \sigma_t)}{C_A(g_t, \sigma_t)} \geq \frac{C_P(s_i, \sigma_i)}{C_A(g_i, \sigma_i)} \right].$$

Then,

$$\begin{aligned} & \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma_1 \cdot \dots \cdot \sigma_n)}{\text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma_1 \cdot \dots \cdot \sigma_n)} \\ = & \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s_1, \sigma_1) + \dots + \text{COST}_{\mathcal{M}}(\mathcal{A}, s_n, \sigma_n)}{\text{opt}_{\mathcal{M}}(g_1, \sigma_1) + \dots + \text{opt}_{\mathcal{M}}(g_n, \sigma_n)} \\ \leq & \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, c_t, \sigma_t)}{\text{opt}_{\mathcal{M}}(g_t, \sigma_t)} \leq R_m. \end{aligned}$$

□

For each state  $s \in S_{\mathcal{A}}(\mathcal{M})$ , let  $\text{maxout}(s)$  denote the maximal weight over all edges, in  $G_{\mathcal{A}}(\mathcal{M})$ , exiting state  $s$ .

**Theorem 5** *Let  $\mu$  equal the distance between the closest two points in  $M$ . Then, the upper bound on the competitive factor of  $P$  is*

$$\frac{2\Psi \cdot \max\{\text{maxout}(s) \mid s \in S_{\mathcal{A}}(\mathcal{M})\}}{\mu}$$

PROOF. From Lemma 4 it follows that a bound on the competitive factor of a space bounded algorithm can be computed simply by considering all the request sequences in  $\psi_1$  defined above. By definition of  $\psi_1$ , for servicing each of these request sequences,  $\sigma$  (in  $\psi_1$  as witnessed by configuration  $g$ ), the adversary moves exactly once, and hence,  $\text{opt}_{\mathcal{M}}(g, \sigma) \geq \mu$ . Since  $G_{\mathcal{A}}(\mathcal{M})$  is finite by definition, there is an absolute largest distance between any two states. This distance is  $\max\{\text{maxout}(s) \mid s \in$

$G_{\mathcal{A}}(\mathcal{M})\}$ . By Lemma 3, we know that while servicing  $\sigma$ , starting at any state  $s$ , the algorithm will not move more than  $2\Psi$  times. Therefore, for any starting state  $s$ ,

$$\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma) \leq 2\Psi \cdot \max\{\text{maxout}(s') \mid s' \in S_{\mathcal{A}}(\mathcal{M})\}.$$

Hence, for any  $\sigma$  and  $s$ ,

$$\begin{aligned} \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma)}{\text{opt}_{\mathcal{M}}(\mathcal{C}(s), \sigma)} &\leq \frac{\text{COST}_{\mathcal{M}}(\mathcal{A}, s, \sigma)}{\text{opt}_{\mathcal{M}}(g, \sigma)} \\ &\leq \frac{2\Psi \cdot \max\{\text{maxout}(s') \mid s' \in G_{\mathcal{A}}(\mathcal{M})\}}{\mu} \end{aligned}$$

□

## 5 *LRU* is Linearly Competitive

Theorem 3 asserts that *LRU* is a competitive paging strategy. Using our results from Section 4.2, we show below that, in fact, the competitive factor of *LRU* is linear in  $k$ , the number of “servers” (pages) used.

Theorem 5 shows that an upper bound on the competitive factor of a space-bounded server algorithm can be found by looking at a combination of the convergence factor of the algorithm and the geometry of the metric space the algorithm is operating in. This offers a wide range of competitive factors among various algorithms. Clearly, the worst algorithm can allow the servers to “wander” no more than every vertex in the digraph before converging. Therefore, an upper bound we could establish for the competitive factor of such an algorithm is

$$\frac{\sum_{s \in S_{\mathcal{A}}(\mathcal{M})} \text{maxout}(s)}{\mu}.$$

In fact, this is the absolute largest upper bound possible for any algorithm. However, it is also not hard to create much better algorithms. Suppose that the player’s server configuration contains the last  $k$  distinct points requested. Clearly the adversary can not remain in the same position for more than  $k$  requests, i.e., the convergence factor of this algorithm equals  $k$ . Therefore, there exist algorithms with convergence factor  $k$ , and their competitive factors are bounded above by

$$\frac{2k \cdot \max\{\text{maxout}(s) \mid s \in S_{\mathcal{A}}(\mathcal{M})\}}{\mu}$$

Consider the worst  $\max\{\text{maxout}(s) \mid s \in S_{\mathcal{A}}(\mathcal{M})\}$ . Clearly, the algorithm must only move one server per request. If it moved more, the new configuration would not reflect the last  $k$  distinct points requested. Therefore, exactly one of the player’s servers may move the largest distance,  $F$ , in the metric space. Thus the worst  $\max\{\text{maxout}(s) \mid s \in S_{\mathcal{A}}(\mathcal{M})\} = F$ .

Thus, the upper bound can be rewritten as

$$\frac{2k \cdot F}{\mu}$$

Consider running the algorithm in a uniform metric space, where  $F = \mu$ . This means that the upper bound of an algorithm with convergence factor  $k$ , operating in a uniform metric space has an upper bound on the competitive factor of  $\mathbf{O}(k)$ . Note that the above algorithm is essentially the well-known paging strategy *LRU*. [2] notes that *LRU* satisfies the necessary and sufficient conditions of Theorem 1 above and hence is competitive in finite metric spaces. Furthermore, the problem of paging can be modeled as a  $k$ -server problem in a finite, uniform metric space [14]. Thus we have that an upper bound on the competitive factor of the *LRU* strategy used for paging is  $\mathbf{O}(k)$ .

## References

- [1] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line algorithms. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 203–210, 1992.
- [2] Ganesh R. Baliga and Anil M. Shende. On space bounded server algorithms. In *Proceedings of the 5th International Conference on Computing and Information*, pages 77–81, 1993.
- [3] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *STOC*, pages 379–386, 1990.
- [4] Marshall Bern, Daniel H. Greene, Arvind Raghunathan, and Madhu Sudan. Online algorithms for locating checkpoints. In *STOC*, pages 359–368, 1990.
- [5] 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.
- [6] 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.
- [7] Marek Chrobak and Lawrence L. Larmore. A new approach to the server problem. *SIAM Journal on Discrete Mathematics*, 4(3):323–328, August 1991.
- [8] Marek Chrobak and Lawrence L. Larmore. On fast algorithms for two servers. *Journal of Algorithms*, pages 203–208, 1991.
- [9] 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.

- [10] 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.
- [11] Don Coppersmith, Peter Doyle, Prabhakar Raghavan, and Marc Snir. Random walks on weighted graphs, and applications to on-line algorithms. In *STOC*, pages 369–378, 1990.
- [12] A. Fiat, Y. Rabani, and Y. Ravid. Competitive  $k$ -server algorithms. In *Proceedings of the 22nd Symposium on Theory of Algorithms*, 1990.
- [13] E. Grove. The harmonic on-line  $k$ -server algorithm is competitive. Manuscript.
- [14] Sandy Irani, Anna R. Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 228–236, 1992.
- [15] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 244–254, 1986.
- [16] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.
- [17] Elias Koutsoupias and Christos Papadimitriou. On the  $k$ -server conjecture. In *Proceedings of the Symposium on Theory of Computing*, pages 507–511, 1994.
- [18] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 322–333, 1988.
- [19] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of the list update and paging rules. *Communications of the ACM*, 28(2):202–208, February 1985.
- [20] Robert Endre Tarjan. Amortized computational complexity. *SIAM Journal on Alg. Disc. Meth.*, 6(2):306–318, April 1985.