# MAX-PLANCK-INSTITUT FÜR INFORMATIK

## New On-Line Algorithms for the Page Replication Problem

Susanne Albers and Hisashi Koga

MPI–I–94–106 February 1994


Im Stadtwald
66123 Saarbrücken
Germany

# New On-Line Algorithms for the Page Replication Problem

Susanne Albers and Hisashi Koga

MPI–I–94–106 February 1994

# New On-Line Algorithms for the Page Replication Problem

Susanne Albers
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Hisashi Koga
The University of Tokyo
Tokyo, Japan

**Abstract**

The page replication problem arises in the memory management of large multiprocessor systems. Given a network of processors, each of which has its local memory, the problem consists of deciding which local memories should contain copies of pages of data so that a sequence of memory accesses can be accomplished efficiently. We present new competitive on-line algorithms for the page replication problem and concentrate on important network topologies for which algorithms with a constant competitive factor can be given. We develop the first optimal randomized on-line replication algorithm for trees and uniform networks; its competitive factor is approximately 1.58. Furthermore we consider on-line replication algorithms for rings and present general techniques that transform large classes of $c$-competitive algorithms for trees into $2c$-competitive algorithms for rings. As a result we obtain a randomized on-line algorithm for rings that is 3.16-competitive. We also derive two 4-competitive on-line algorithms for rings which are either deterministic or memoryless. All our algorithms improve the previously best competitive factors for the respective topologies.

## 1 Introduction

This paper deals with problems that arise in the memory management of large multiprocessor systems. Such multiprocessing environments typically consist of a network of processors, each of which has its local memory. A global shared memory is modeled by distributing the physical pages among the local memories. Accesses to the global memory are then accomplished by accessing the local memories. Suppose a processor $p$ wants to read a memory address from page $A$. If $A$ is stored in $p$'s local memory, then this read operation can be accomplished locally. Otherwise, $p$ determines a processor $q$ holding the page and sends a request to $q$. The desired information is then transmitted from $q$ to $p$, and the communication cost incurred thereby is proportional to the distance from $q$ to $p$. If $p$ has to access page $A$ frequently, it may be worthwhile to move or copy $A$ from $q$ to $p$ because subsequent accesses will become cheaper. However, transmitting an entire page incurs a high communication cost proportional to the page size times the distance from $q$ to $p$.

If a page is writable, it is reasonable to store only one copy of the page in the entire system. This avoids the problem of keeping multiple copies of the page consistent. The *migration problem* is to decide in which local memory the single copy of the page should be stored so that a sequence of memory accesses can be processed at low cost. On the other hand, if a page is read-only, it is possible to keep several copies of the page in the system, i.e. a page may be copied from one

local memory to another. In the *replication problem* we have to determine which local memories should contain copies of the page. Finding efficient migration and replication strategies is an important problem that has been studied from both a practical and theoretical point of view [DF82, SD89, BS89, BFR92, W92, ABF93, CLRW93, K93]. In this paper we will study on-line algorithms for the page replication problem. In order to analyze the performance of an on-line algorithm we will use *competitive analysis* [ST85], the worst case ratio of cost incurred by an on-line algorithm and the cost incurred by an optimal off-line algorithm.

Awerbuch *et al.* [ABF93] have presented a deterministic on-line replication strategy for general graphs that achieves an optimal competitive ratio of $O(\log n)$, where $n$ is the number processors. However, for many important topologies, this bound is not very expressive. Black and Sleator [BS89], who have initiated the theoretical study of the replication problem, proposed a 2-competitive deterministic on-line algorithm for trees and uniform networks. A uniform network is a complete graph in which all edges have the same length. Black and Sleator also proved that no deterministic on-line replication algorithm can be better than 2-competitive. Recently Koga [K93] has developed a randomized on-line replication algorithm for trees that is 1.71-competitive for large values of the page size, thereby beating the deterministic lower bound. He also presented a randomized 4-competititive algorithm for the case that the network topology forms a ring. However, his algorithm uses a large amount of randomness, namely one random number for each read operation. The competitive ratios hold against the oblivious adversary [BBKTW94]. Bartal *et al.* [BFR92] have presented a randomized replication algorithm for rings which is $2(2+\sqrt{3})$-competitive against adaptive adversaries. Using the 4-comptitive algorithm by Koga and the $2(2+\sqrt{3})$-competitive algorithm by Bartal *et al.*, one can construct a deterministic replication algorithm for the ring which achieves a competitive ratio of $4 \cdot 2(2+\sqrt{3}) \approx 29.86$, see [BBKTW94] for details. However, that algorithm is very complicated and not useful in practical applications.

In this paper we develop a number of new deterministic and randomized on-line replication algorithms. We concentrate on network topologies that are important in practice and for which on-line algorithms with a constant competitive factor can be developed. In Section 4 we present a randomized on-line replication algorithm for trees and uniform networks, called GEOMETRIC, which is $(\frac{\rho^r}{\rho^r-1})$-competitive. Here $\rho = \frac{r+1}{r}$ and $r$ is the page size factor. For large values of $r$, which occur in practice, GEOMETRIC's competitiveness is approximately $\frac{e}{e-1} \approx 1.58$. We also show that GEOMETRIC is optimal. Specifically, we prove that no randomized on-line replication algorithm can be better than $(\frac{\rho^r}{\rho-1})$-competitive. Interestingly, our algorithm GEOMETRIC uses only one random number during an initialization phase and runs completely deterministically thereafter. Such algorithms which use only a very little amount of randomness are valuable from a practical standpoint because random bits are usually an expensive resource. In Section 5 we consider replication algorithms for rings. We present a deterministic technique that transforms a large class of $c$-competitive algorithms for trees into $2c$-competitive algorithms for rings. As a result we obtain a randomized $(\frac{2\rho^r}{\rho^r-1})$-competitive algorithm for rings that also uses only one random number during an initialization phase. Note that the competitive ratio is approximately 3.16 and beats the previously best ratio of 4. We also derive two 4-competitive algorithms for rings which are either deterministic or memoryless. Our 4-competitive deterministic algorithm greatly improves the competitive factor of 29.86 men-

tioned above. Furthermore, our algorithm is very simple, as opposed to the 29.86-competitive algorithm. Finally we present a randomized version of our deterministic technique for constructing ring algorithms and prove that this randomized variant achieves the same performance. All our randomized competitive factors hold against the oblivious adversary.

## 2 Problem statement and competitive analysis

Formally, the page replication problem can be described as follows. We are given an undirected graph $G$. Each node in $G$ corresponds to a processor and the edges represent the interconnection network. Associated with each edge is a *length* that is equal to the distance between the connected processors. We assume that the edge lengths satisfy the triangle inequality. In the page replication problem we generally concentrate on one particular page. We say that *a node $v$ has the page* if the page is contained in $v$'s local memory. *A request at a node $v$* occurs if $v$ wants to read an address from the page. The request can be satisfied at zero cost if $v$ has the page. Otherwise the request is served by accessing a node $w$ holding the page and the incurred cost equals the distance from $v$ to $w$. Immediately after the request, the page may be replicated into $v$'s local memory. The cost incurred by this replication is $r$ times the distance from $v$ to $w$. Here $r$ denotes the page size factor. In practical applications, $r$ is a large value, usually several hundred or thousand. (The page may only be replicated *after* a request because it is impossible to delay the service of a memory access while the entire page is copied.) We study the page replication problem under the assumption that a node having the page never drops it. A page replication algorithm is usually presented with an entire sequence of requests that must be served with low total cost. A page replication algorithm is *on-line* if it serves every request without knowledge of any future requests.

We analyze the performance of on-line page replication algorithms using *competitive analysis* [ST85]. In a competitive analysis, the cost incurred by an on-line algorithm is compared to the cost incurred by an *optimal off-line algorithm*. An optimal off-line algorithm knows the entire request sequence in advance and can serve it with minimum cost. Let $C_A(\sigma)$ and $C_{OPT}(\sigma)$ be the cost of the on-line algorithm $A$ and the optimal off-line algorithm OPT on request sequence $\sigma$. Usually an on-line algorithm $A$ is called $c$-competitive if there exists an constant $a$ such that for every request sequence

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + a.$$

However, this definition is not reasonable in the context of page replication because an on-line replication algorithm can be 0-competitive by replicating the page initially to all processors and assigning that cost to the constant $a$. Therefore, we call an on-line replication algorithm $A$ $c$-competitive if

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma)$$

for all request sequences $\sigma$. If $A$ is a randomized algorithm, then $C_A(\sigma)$ must be replaced by the expected cost incurred by $A$, where the expectation is taken over the random choices made by $A$. In this paper we evaluate randomized on-line algorithms only against the *oblivious adversary*, see [BBKTW94]. The *competitive factor* of an on-line algorithm $A$ is the infimum of all $c$ such that $A$ is $c$-competitive.

## 3 Basic definitions and techniques

A substantial part of this paper deals with on-line replication algorithms for trees. Even when considering uniform networks and rings, we will reduce the algorithms and their analyses to the case that the underlying topology forms a tree. For this reason we introduce some basic definitions for trees.

Consider an arbitrary tree. The root of the tree is generally denoted by $s$. We assume that initially, only $s$ has the page. Consider an undirected edge $e = \{v, w\}$ in the tree. The node in $\{v, w\}$ that is farther away from the root is called the *child node* of $e$. The length of $e$ is denoted by $l(e)$.

In the following we will always assume that if an algorithm (on-line or off-line) replicates the page from a node $v$ to a node $w$, then the page is also replicated to all nodes on the path from $v$ to $w$. This does not incur extra cost. Thus, the nodes with the page always form a connected component of the given tree. Note that if a node $v$ does not have the page, then the closest node $w$ with the page lies on the path from $v$ to the root, and all paths from $v$ to a node with the page pass through $w$. Therefore, we may assume without loss of generality that a replication algorithm always serves requests at a node not holding the page by accessing the closest node with the page. This cannot increase the total cost incurred in serving the whole request sequence.

We present a technique that we will frequently use to analyze on-line replication algorithms for trees. Let $T$ be a tree and $\sigma$ be a request sequence for $T$. Let OPT be the optimal off-line replication algorithm. We usually analyze an on-line replication algorithm $A$ by partitioning the cost that is incurred by A and by OPT into parts that are incurred by each edge of the tree. Suppose an algorithm serves a request at a node $v$. Then an edge $e$ incurs a cost equal to the length of $e$ if $e$ belongs to the path from $v$ to the closest node with the page. If $e$ does not belong to that path, then $e$ incurs a cost of zero. An edge also incurs the cost of a replication across it. Given an arbitrary tree $T$ and a request sequence $\sigma$ for $T$, let $C_A(\sigma, e)$ denote the cost that is incurred by edge $e$ when $A$ serves $\sigma$. Analogously, let $C_{OPT}(\sigma, e)$ be the cost that is incurred by $e$ when OPT serves $\sigma$. (If $A$ is a randomized algorithm, then $C_A(\sigma, e)$ is the expected cost incurred by $e$, where the expectation is taken over the random choices made by $A$.) We generally evaluate the performance of an on-line algorithm $A$ by comparing $C_A(\sigma, e)$ to $C_{OPT}(\sigma, e)$ for all edges $e$ of the tree.

In order to analyze $C_A(\sigma, e)$, we use some notation. Let $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ be a request sequence of length $m$ and let $\sigma(t)$, $1 \leq t \leq m$, be the request at time $t$. Suppose $\sigma(t)$ is a request at node $v$. We set

$$a_\sigma(e, \sigma(t)) = 1$$

if $e$ belongs to the path from $v$ to the root. Otherwise we set

$$a_\sigma(e, \sigma(t)) = 0.$$

If $a_\sigma(e, \sigma(t)) = 1$, we say that $\sigma(t)$ *causes an access at edge* $e$. Let

$$a_\sigma(e) = \sum_{t=1}^{m} a_\sigma(e, \sigma(t)),$$

i.e. $a_\sigma(e)$ is the number of requests that cause an access at edge $e$. The following simple lemma is crucial in our analyses.

**Lemma 1** *Let $A$ be an on-line replication algorithm that, given an arbitrary tree $T$ and a request sequence $\sigma$ for $T$, satisfies*

$$C_A(\sigma, e) \leq c \cdot \min\{a_\sigma(e), r\} \cdot l(e)$$

*for all edges $e$. Then the algorithm $A$ is $c$-competitive. (Again, if $A$ is a randomized algorithm, then $C_A(\sigma, e)$ is the expected cost incurred by $e$.)*

**Proof:** We prove that for any edge $e$, $C_A(\sigma, e) \leq c \cdot C_{OPT}(\sigma, e)$. This implies the lemma.

If $a_\sigma(e) < r$, then OPT does not replicate the page across $e$ and $e$ incurs a cost of $a_\sigma(e)l(e)$. Hence

$$C_A(\sigma, e) \leq c \cdot \min\{a_\sigma, r\}l(e) = c \cdot a_\sigma(e) \cdot l(e) = c \cdot C_{OPT}(\sigma, e).$$

On the other hand, if $a_\sigma(e) \geq r$, then OPT replicates the page across $e$, and $e$ incurs a cost of $rl(e)$. Thus

$$C_A(\sigma, e) \leq c \cdot \min\{a_\sigma, r\}l(e) = c \cdot r \cdot l(e) = c \cdot C_{OPT}(\sigma, e). \quad \square$$

# 4 An optimal algorithm for trees and uniform networks

First we will describe and analyze a randomized on-line algorithm for trees. This algorithm can be applied to uniform networks, too. Then we prove that the competitive factor of our algorithm is optimal for all values of $r$. Throughout this section let $\rho = \frac{r+1}{r}$.

**Algorithm GEOMETRIC (for trees):** The algorithm first chooses a random number from the set $\{1, 2, \ldots, r\}$. Specifically, the number $i$ is chosen with probability $p_i = \alpha \cdot \rho^{i-1}$, where $\alpha = \frac{\rho - 1}{\rho^r - 1}$. While processing the request sequence, the algorithm maintains a count on each edge of the tree. Initially, all counts are set to 0. If there is a request at a node $v$ that does not have the page, then all counts along the path from $v$ to the closest node with the page are incremented by 1. When a count reaches the value of the randomly chosen number, the page is replicated to the child node of the corresponding edge.

Before we analyze the performance of GEOMETRIC, we mention a few observations and remarks. The algorithm is called GEOMETRIC because $p_{i+1}/p_i = \rho$ is constant for all $i = 1, 2, \ldots, r-1$. Note that

$$\sum_{i=1}^{r} p_i = \alpha \sum_{i=1}^{r} \rho^{i-1} = \alpha \cdot \frac{\rho^r - 1}{\rho - 1} = \alpha \cdot \frac{1}{\alpha} = 1.$$

Suppose that GEOMETRIC processes a request sequence $\sigma$. It is easy to prove by induction on the number of requests processed so far that the counts on a path from the root to a node $v$ are monotonically non-increasing. Furthermore, after each request, a node has the page if and only if it is the child node of an edge whose count is equal to the value of the randomly chosen number.

**Theorem 1** *For any tree, the algorithm GEOMETRIC is $(\frac{\rho^r}{\rho^r - 1})$ – competitive.*

Note that $(\frac{\rho^r}{\rho^r - 1})$ goes to $\frac{e}{e-1} \approx 1.58$ as $r$ tends to infinity. Furthermore, GEOMETRIC uses only one random number during an initialization phase and runs completely deterministically thereafter.

The proof of the theorem follows from Lemma 1 and Lemma 2 below. For a given tree $T$ and an arbitrary request sequence $\sigma$ on $T$, let $E[C_G(\sigma, e)]$ denote the expected cost incurred by edge $e$ when GEOMETRIC serves $\sigma$.

**Lemma 2** *Given an arbitrary tree $T$ and request sequence $\sigma$ for $T$, the algorithm GEOMETRIC satisfies*

$$E[C_G(\sigma, e)] \leq (\frac{\rho^r}{\rho^r - 1}) \cdot \min\{a_\sigma(e), r\} \cdot l(e)$$

*for all edges $e$ of $T$.*

**Proof:** Consider an arbitrary tree $T$ and a request sequence $\sigma$ for $T$. Let $e$ be an edge of the tree. Furthermore, let $k = a_\sigma(e)$ and $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$ be the requests that cause an access at the edge $e$. Note that the algorithm GEOMETRIC increases the count of $e$ exactly at the requests $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$, provided that the page has not been replicated across $e$ so far.

First, assume that $k > r$. Since $\sum_{i=1}^{r} p_i = 1$, GEOMETRIC has replicated the page across $e$ before request $\sigma(t_{r+1})$. Thus the edge $e$ incurs the same cost as if we had $k = r$. For this reason it suffices to consider the case that $k$ satisfies $1 \leq k \leq r$ and show

$$E[C_G(\sigma, e)] \leq c \cdot k \cdot l(e), \tag{1}$$

where $\frac{\rho^r}{\rho^r - 1}$. This proves the lemma.

So suppose we have $1 \leq k \leq r$. The algorithm GEOMETRIC first chooses a random number $i$ from the set $\{1, 2, \ldots, r\}$. If $i$ satisfies $i \leq k$, the edge $e$ incurs a cost of $r + i$. Otherwise $e$ incurs a cost of $k$. Thus

$$\begin{aligned}
E[C_G(\sigma, e)] &= l(e)(\sum_{i=1}^{k}(r+i)p_i + \sum_{i=k+1}^{r} kp_i) \\
&= l(e)(\sum_{i=1}^{k} r\alpha\rho^{i-1} + \sum_{i=1}^{k} i\alpha\rho^{i-1} + \sum_{i=k+1}^{r} k\alpha\rho^{i-1}) \\
&= \alpha l(e)(\frac{r(\rho^k - 1)}{\rho - 1} + \frac{k\rho^{k+1} - (k+1)\rho^k + 1}{(\rho-1)^2} + \frac{k(\rho^r - 1) - k(\rho^k - 1)}{\rho - 1}).
\end{aligned}$$

We have $\rho - 1 = \frac{1}{r}$. Thus

$$\begin{aligned}
E[C_G(\sigma, e)] &= \frac{\alpha l(e)}{\rho - 1}(r(\rho^k - 1) + k\rho^k - r(\rho^k - 1) + k(\rho^r - \rho^k)) \\
&= \frac{\alpha l(e)}{\rho - 1}(k\rho^r) \\
&= \frac{\rho^r}{\rho^r - 1} \cdot k \cdot l(e)
\end{aligned}$$

and this proves inequality (1). □

The algorithm GEOMETRIC is easily applied to uniform networks. Consider an arbitrary uniform network and let $s$ be the node that has the page initially. Since all edges in the graph have the same length, we may assume without loss of generality that a replication algorithm

(on-line or off-line) serves requests and replicates the page only along edges $\{s, v\}$. Hence the network can be reduced to a tree by neglecting the edges $\{v, w\}$ with $s \neq v$, $s \neq w$. Run on this tree, the algorithm GEOMETRIC is $(\frac{\rho^r}{\rho^r-1})$-competitive.

We now prove that GEOMETRIC's competitive factor is optimal for all values of $r$.

**Theorem 2** *Let $A$ be a randomized on-line replication algorithm. Then $A$ cannot be better than $(\frac{\rho^r}{\rho^r-1})$-competitive, even on a graph consisting of two nodes.*

**Proof of Theorem 2:** Let $s$ and $t$ be two nodes that connected by an edge of length 1. We assume that initially, only node $s$ has the page. We will construct a request sequence $\sigma$ consisting of requests at node $t$ such that the expected cost incurred by $A$ is at least $\frac{\rho^r}{\rho^r-1}$ times the optimal off-line cost.

For $i = 1, 2, \ldots$, let $q_i$ be the probability that $A$ replicates the pages from $s$ to $t$ after exactly $i$ requests, given a request sequence that consists only of requests at node $t$. In the following we compare the algorithm $A$ to the algorithm GEOMETRIC. Let $E[C_A(\sigma)]$ and $E[C_G(\sigma)]$ denote the expected cost incurred by $A$ and GEOMETRIC on a request sequence $\sigma$. Furthermore, for $i = 1, 2, \ldots, r$, let $p_i = \alpha \cdot \rho^{i-1}$. We consider two cases.

Case 1: There exists an $l$, where $1 \leq l \leq r$, such that $\sum_{i=1}^{l} q_i \geq \sum_{i=1}^{l} p_i$.
Let $k$ be the smallest number satisfying the above inequality, i.e. $\sum_{i=1}^{k} q_i \geq \sum_{i=1}^{k} p_i$ and $\sum_{i=1}^{j} q_i < \sum_{i=1}^{j} p_i$ for all $j$ with $1 \leq j < k$. Let $\sigma$ be the request sequence that consists of $k$ requests at node $t$. We show that

$$E[C_A(\sigma)] - E[C_G(\sigma)] \geq 0. \tag{2}$$

Calculating $E[C_G(\sigma)]$ in the same way as in the proof of Lemma 2, inequality (2) implies $E[C_A(\sigma)] \geq E[C_G(\sigma)] = \frac{\rho^r}{\rho^r-1} \cdot k$. Hence $A$ cannot be better than $(\frac{\rho^r}{\rho^r-1})$-competitive because the optimal off-line cost on $\sigma$ equals $k$. We have

$$E[C_A(\sigma)] = \sum_{i=1}^{k}(r+i)q_i + \sum_{i \geq k+1} k q_i = \sum_{i=1}^{k}(r+i)q_i + k(1 - \sum_{i=1}^{k} q_i)$$

$$E[C_G(\sigma)] = \sum_{i=1}^{k}(r+i)p_i + \sum_{i \geq k+1} k p_i = \sum_{i=1}^{k}(r+i)p_i + k(1 - \sum_{i=1}^{k} p_i).$$

Hence

$$\begin{aligned} E[C_A(\sigma)] - E[C_G(\sigma)] &= \sum_{i=1}^{k}(r+i)(q_i - p_i) + k\sum_{i=1}^{k}(p_i - q_i) \\ &= \sum_{i=1}^{k} i(q_i - p_i) + (r-k)\sum_{i=1}^{k}(q_i - p_i) \end{aligned}$$

Since $\sum_{i=1}^{k} q_i \geq \sum_{i=1}^{k} p_i$ and $r - k \geq 0$, we obtain

$$E[C_A(\sigma)] - E[C_G(\sigma)] \geq \sum_{i=1}^{k} i(q_i - p_i) = \sum_{i=1}^{k}(\sum_{j=i}^{k} q_i - \sum_{j=i}^{k} p_i).$$

For $i = 2, 3, \ldots, k$ we have $\sum_{j=1}^{i-1} q_i < \sum_{j=1}^{i-1} p_i$ and hence

$$\sum_{j=i}^{k} q_i - \sum_{j=i}^{k} p_i > \sum_{j=i}^{k} q_i - \sum_{j=i}^{k} p_i + \sum_{j=1}^{i-1} q_i - \sum_{j=1}^{i-1} p_i = \sum_{j=1}^{k} q_i - \sum_{j=1}^{k} p_i.$$

We conclude

$$E[C_A(\sigma)] - E[C_G(\sigma)] \geq \sum_{i=1}^{k}(\sum_{j=i}^{k} q_i - \sum_{j=i}^{k} p_i) \geq \sum_{i=1}^{k}(\sum_{j=1}^{k} q_i - \sum_{j=1}^{k} p_i) \geq 0$$

and inequality (2) is proved.

Case 2: For all $k = 1, 2, \ldots, r$, the inequality $\sum_{i=1}^{k} q_i < \sum_{i=1}^{k} p_i$ is satisfied.
Let $\sigma$ be the request sequence that consists of $2r$ requests at node $t$. Let $A'$ be the on-line algorithm with $q'_i = q_i$, for $i = 1, 2, \ldots, r-1$, and $q'_r = \sum_{i \geq r} q_i$. Then

$$E[C_A(\sigma)] = \sum_{i=1}^{2r}(r+i)q_i + \sum_{i>2r} 2rq_i \geq \sum_{i=1}^{r-1}(r+i)q_i + 2rq'_r = \sum_{i=1}^{r-1}(r+i)q'_i + 2rq'_r = E[C_{A'}(\sigma)].$$

Since $\sum_{i=1}^{r} q'_i = \sum_{i=1}^{r} p_i = 1$ and $\sum_{i=1}^{j} q'_i < \sum_{i=1}^{j} p_i$ for all $j$ with $1 \leq j < r$, Case 1 immediately implies

$$E[C_A(\sigma)] \geq E[C_{A'}(\sigma)] \geq E[C_G(\sigma)] = \frac{\rho^r}{\rho^r - 1} r,$$

and $A$ cannot be better than $(\frac{\rho^r}{\rho^r - 1})$-competitive because the optimal off-line cost equals $r$. □

# 5 Algorithms for the ring

In this section we assume that the given net of processors forms a ring. We will present techniques that transforms a large class of $c$-competitive algorithms for trees into $2c$-competitive algorithms for rings.

We assume that initially, only one node of the ring, say $s$, has the page. Let $n$ be the number of nodes in the ring and let $v_1, v_2, \ldots, v_n$ be the nodes if we scan the ring in clockwise direction starting from $s$, i.e. $v_1 = s$. For $i = 1, 2, \ldots, n$, let $e_i = \{v_i, v_{i+1}\}$ be the undirected edge from $v_i$ to $v_{i+1}$. Naturally, $v_{n+1}$ equals $v_1$. Let $x$ and $y$ be any two points on the ring; $x$ and $y$ need not necessarily be processor nodes. We denote by $(x, y)$ the arc of the ring that is obtained if we start in $x$ and go to $y$ in clockwise direction. Let $l(x, y)$ be the length of the arc $(x, y)$.

**Algorithm RING:** Let $P$, $P \neq s$, be the point on the ring satisfying $l(s, P) = l(P, s)$, i.e. $P$ is the point "opposite" to $s$. The algorithm first cuts the ring at $P$. It regards the resulting structure as a tree $T$ with root $s = v_1$. The arc $(s, P)$ represents one branch of the tree and the arc $(P, s)$ represents another branch of the tree (see Figure 1). We assume that the point $P$ becomes part of the arc $(s, P)$. This is significant if $P$ coincides with one of the nodes $v_i$. The algorithm RING then uses an on-line replication algorithm $A$ for trees in order to serve a request sequence $\sigma$. That is, RING assumes that $\sigma$ is a request sequence for $T$ and serves the request sequence using the tree algorithm $A$.

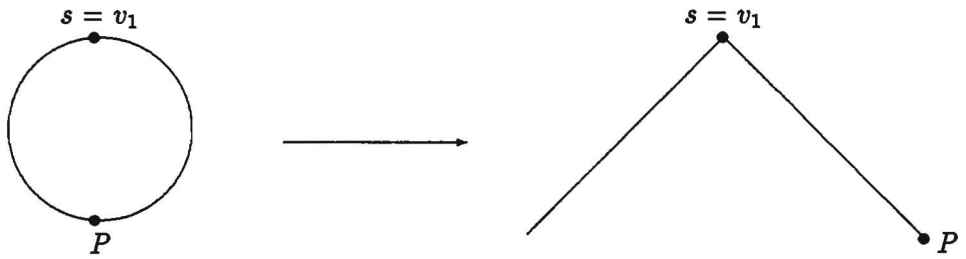



Figure 1: A cut of the ring

**Theorem 3** *Let $A$ be an on-line replication algorithm that, given an arbitrary tree $T$ and a request sequence $\sigma$ for $T$, satisfies*

$$C_A(\sigma, e) \leq c \cdot \min\{a_\sigma(e), r\} l(e) \tag{3}$$

*for all edges $e$ of the tree. ($C_A(\sigma, e)$ is the expected cost incurred by $e$ if $A$ is a randomized algorithm.) If the algorithm RING uses $A$ as tree algorithm, then the resulting algorithm is $2c$-competitive.*

Before we prove this theorem, we mention some important implications. Lemma 2 immediately implies the following result.

**Corollary 1** *If RING uses the algorithm GEOMETRIC as tree algorithm, then the resulting algorithm is $c$-competitive, where $c = \frac{2\rho^r}{\rho^r - 1}$.*

We observe that $c$ goes to $\frac{2e}{e-1} \approx 3.16$ as $r$ tends to infinity. Also note that if RING uses the GEOMETRIC algorithm, then only one random number is used during an initialization phase. While processing a requests sequence, the algorithm runs completely deterministically. Next we consider a deterministic replication algorithm for trees, called DETERMINISTIC_COUNT. This algorithm was proposed by Black and Sleator [BS89] who showed that it achieves an optimal competitive factor of 2.

**Algorithm DETERMINISTIC_COUNT:** The algorithm maintains a count on each edge of the tree. Initially, all counts are set to zero. While a request sequence is processed, the counts are incremented in the same way as by the algorithm GEOMETRIC. However DETERMINISTIC_COUNT does not choose a random number in order to determine when a replication should occur. Rather it replicates the page to the child node of an edge when the corresponding count reaches $r$.

It is easy to see that, given an arbitrary tree $T$ and a request sequence $\sigma$,

$$C_{DC}(\sigma, e) \leq 2 \cdot \min\{a_\sigma(e), r\} \cdot l(e)$$

for all edges $e$. Here $C_{DC}(\sigma, e)$ denotes the cost that is incurred by edge $e$ when DETERMINISTIC_COUNT serves $\sigma$.

**Corollary 2** *If the algorithm RING uses DETERMINISTIC_COUNT as tree algorithm, then the resulting algorithm is 4-competitive.*

We remark that the combination of RING and DETERMINISTIC_COUNT runs completely deterministically. Another interesting on-line replication algorithm for trees was presented by Koga [K93].

**Algorithm COINFLIP:** If there is a request at a node with the page, then the algorithm performs no action. If there is a request at a node $v$ without the page, the algorithm serves the request by accessing the closest node $u$ with the page. Then with probability $\frac{1}{r}$, the algorithm replicates the page from $u$ to $v$.

Theorem 3 and Lemma 3 below imply the following result.

**Corollary 3** *If RING uses the algorithm COINFLIP as tree algorithm, then the resulting algorithm is 4-competitive.*

The combination of RING and COINFLIP is memoryless [RS89], i.e. it does not need any memory (for instance for counts) in order to determine when a replication should take place.

**Lemma 3** *Let $T$ be an arbitrary tree and $\sigma$ be arbitrary request sequence for $T$. For an edge $e$ in $T$, let $E[C_{CF}(\sigma, e)]$ be the expected cost incurred by $e$ when COINFLIP serves $\sigma$. Then*

$$E[C_{CF}(\sigma, e)] \leq 2 \cdot \min\{a_\sigma(e), r\} \cdot l(e)$$

*for all edges $e$ of $T$.*

**Proof:** Given a tree $T$ and a request sequence $\sigma$, consider an arbitrary edge $e$ of $T$. Each time there is a request $\sigma(t)$ with $a_\sigma(e,t) = 1$, the algorithm COINFLIP replicates the page across $e$ with probability $\frac{1}{r}$, provided the replication across $e$ has not occurred so far. Let $k = a_\sigma(e)$ and let $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$ be the requests which cause an access at edge $e$. The probability that the replication of the page across $e$ occurs after exactly $i$ of these requests equals

$$(1 - \frac{1}{r})^{i-1} \cdot \frac{1}{r}.$$

With probability $(1 - \frac{1}{r})^k$, no replication occurs. We have

$$\begin{aligned}
E[C_{CF}(\sigma, e)] &= l(e)(\sum_{i=1}^{k}(r+i)(1-\frac{1}{r})^{i-1}\frac{1}{r} + k(1-\frac{1}{r})^k) \\
&= l(e)(\sum_{i=1}^{k}(1-\frac{1}{r})^{i-1} + \frac{1}{r}\sum_{i=1}^{k} i(1-\frac{1}{r})^{i-1} + k(1-\frac{1}{r})^k) \\
&= l(e)(\frac{1-(1-\frac{1}{r})^k}{1-(1-\frac{1}{r})} + \frac{1}{r}\frac{k(1-\frac{1}{r})^{k+1} - (k+1)(1-\frac{1}{r})^k + 1}{(1-(1-\frac{1}{r}))^2} + k(1-\frac{1}{r})^k) \\
&= l(e)(r(2 + k(1-\frac{1}{r})^{k+1} - (k+2)(1-\frac{1}{r})^k) + k(1-\frac{1}{r})^k) \\
&= l(e)(r(2 + k(1-\frac{1}{r})^k - \frac{k}{r}(1-\frac{1}{r})^k - (k+2)(1-\frac{1}{r})^k) + k(1-\frac{1}{r})^k) \\
&= l(e)(r(2 - 2(1-\frac{1}{r})^k).
\end{aligned}$$

Thus

$$E[C_{CF}(\sigma, e)] = 2l(e)r(1 - (1 - \frac{1}{r})^k).$$

If $k = a_\sigma(e) \geq r$, then

$$E[C_{CF}(\sigma, e)] = 2l(e)r(1 - (1 - \frac{1}{r})^r) \leq 2l(e)r = 2\min\{a_\sigma(e), r\}l(e),$$

and the lemma is proved.

Suppose $k < r$. We show that

$$1 - (1 - \frac{1}{r})^k \leq \frac{k}{r}. \tag{4}$$

For $x \in \mathbb{R}$, let $f_1(x) = 1 - (1 - \frac{1}{r})^x$ and $f_2(x) = \frac{x}{r}$. Note that $f_1(x)$ is an exponential function whereas $f_2(x)$ is a line with slope $\frac{1}{r}$. Hence $f_1(x)$ and $f_2(x)$ can intersect in at most two points. We have $f_1(0) = f_2(0)$ and $f_1(1) = f_2(1)$. Since $f_1(x)$ is bounded from above by 1 and $f_2(x)$ is an unbounded function, we conclude that $f_1(x) \leq f_2(x)$ or all $x \geq 1$ and inequality (4) must hold. Hence

$$E[C_{CF}(\sigma, e)] = 2l(e)r(1 - (1 - \frac{1}{r})^k) \leq 2l(e)r\frac{k}{r} = 2l(e)k = 2\min\{a_\sigma(e), r\} \cdot l(e).$$

This concludes the proof of the lemma. □

Finally we present a randomized variant of the algorithm RING.

**Algorithm RING(RANDOM):** The algorithm works in the same as the algorithm RING. However, instead of cutting the ring at the point opposite to $s$, the algorithm RING(RANDOM) chooses a point $P$ uniformly at random on the ring and cuts the ring at that point $P$.

We can show a statement analogous to Theorem 3.

**Theorem 4** *Let $A$ be an on-line replication algorithm that, given an arbitrary tree $T$ and a request sequence $\sigma$ for $T$, satisfies*

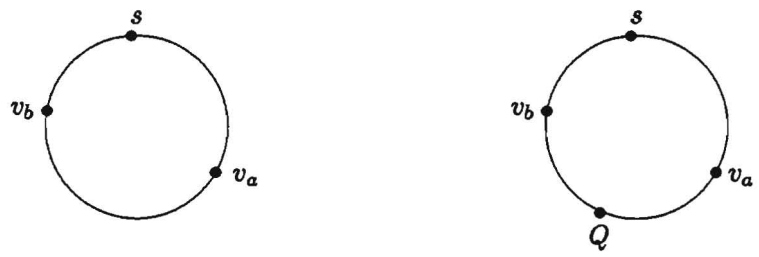
$$C_A(\sigma, e) \leq c \cdot \min\{a_\sigma(e), r\}l(e) \tag{5}$$

*for all edges $e$ of the tree. ($C_A(\sigma, e)$ is the expected cost incurred by $e$ if $A$ is a randomized algorithm.) If the algorithm RING(RANDOM) uses $A$ as tree algorithm, then the resulting algorithm is $2c$-competitive.*

Theorem 4 implies that statements analogous to Corollaries 1 - 3 hold. Note, however, that a combination of RING(RANDOM) and DETERMINISTIC_COUNT is not a purely deterministic algorithm.

It remains to prove the two main theorems.

**Proof of Theorem 3:** Let $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ be a request sequence for the ring. We start with some observations on how OPT serves $\sigma$. Consider the state of the ring after OPT has served $\sigma$. Let $v_a$ be the node farthest from $s$ to which OPT has replicated the page in clockwise direction. Similarly, let $v_b$ be the node farthest from $s$ to which OPT has replicated the page in counter-clockwise direction. Figure 2(a) illustrates this situation.




(a) (b)

Figure 2: The state of the ring when OPT serves $\sigma$.

We may assume without loss of generality that OPT replicates the page from $s$ to $v_a$ and from $s$ to $v_b$ at the beginning of the request sequence, before any requests are served. This does not incur a higher cost as if the replication is done while requests are processed. Any request at a node that belongs to $(s, v_a)$ or $(v_b, s)$ can then be served at zero cost. Let $Q$ be the point on $(v_a, v_b)$ which satisfies $l(v_a, Q) = l(Q, v_b)$, see Figure 2(b). Any request at a node $v_i$ that belongs to $(v_a, Q)$ is served by accessing $v_a$ and the incurred cost equals $l(v_a, v_i)$. Any request at a node $v_i$ that belongs to $(Q, v_b)$ is served by accessing $v_b$, and the incurred cost equals $l(v_i, v_b)$. Let $I = \{1, 2, \ldots, n\}$. Let $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$ be the requests in $\sigma$ which request a node that does not belong to $(s, v_a)$ or $(v_b, s)$. For $j = 1, 2, \ldots, k$, let $v_{\mu(t_j)}$ be the node requested by $\sigma(t_j)$. The cost incurred by OPT in serving $\sigma$ equals

$$C_{OPT}(\sigma) = rl(s, v_a) + rl(v_b, s) + \sum_{j=1}^{k} \min\{l(v_a, v_{\mu(t_j)}), l(v_{\mu(t_j)}, v_b)\}.$$

Note that also OPT implicitely uses a tree in order to serve the requests sequence. This tree is obtained if the ring is cut at the point $Q$. Let $C_R(\sigma)$ be the cost incurred by RING in serving $\sigma$. In the following we show that

$$C_R(\sigma) \leq 2c \cdot C_{OPT}(\sigma). \tag{6}$$

This implies the theorem.

For the analysis of $C_R(\sigma)$ we need some more notation. Let $T$ be the tree that is obtained if the ring is cut at point $P$. Let $i \in \{1, 2, \ldots, n\}$ and $t \in \{1, 2, \ldots, m\}$ be arbitrary. We denote by $v_{\mu(t)}$ the node requested at time $t$. We set

$$a_\sigma(e_i, t) = 1$$

if in the tree $T$, $e_i$ is on the path from $v_{\mu(t)}$ to the root. Otherwise we set $a_\sigma(e_i, t) = 0$. If $a_\sigma(e_i, t) = 1$, we say that $\sigma(t)$ *causes an access at the edge* $e_i$ *in the tree* $T$. We set

$$a_\sigma(e_i) = \sum_{t=1}^{m} a_\sigma(e_i, t).$$

By inequality (3), $RING$ incurs a total cost of

$$C_R(\sigma) = \sum_{j=1}^{n} c \min\{a_\sigma(e_i), r\} l(e_i).$$


Figure 3: Point $P$ belongs to $(s, v_a)$ or $(v_b, s)$

In the following we investigate two cases. First we will consider the case that $P$ belongs either to $(s, v_a)$ or to $(v_b, s)$. Then we will study the case that $P$ belongs neither to $(s, v_a)$ nor to $(v_b, s)$.

Case 1: Suppose that $P$ belongs either to $(s, v_a)$ or to $(v_b, s)$, see Figure 3.
Then

$$\sum_{i \in I} l(e_i) \leq 2 \max\{l(s, v_a), l(v_b, s)\} \leq 2(l(s, v_a) + l(v_b, s))$$

and

$$\begin{aligned} C_R(\sigma) &\leq \sum_{i \in I} c \min\{a_\sigma(e_i), r\} l(e_i) \\ &\leq cr \sum_{i \in I} l(e_i) \\ &\leq 2cr(l(s, v_a) + l(v_b, s)) \\ &\leq 2c \cdot C_{OPT}(\sigma). \end{aligned}$$

Inequality (6) is proved.

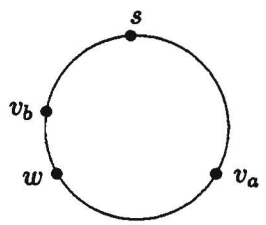



Figure 4: The location of $w$

Case 2: Now suppose that $P$ belongs neither to $(s, v_a)$ nor to $(v_b, s)$.
We only consider the case that $l(s, v_a) \geq l(v_b, s)$. The case $l(v_b, s) \geq l(s, v_a)$ is symmetric. Let $w$ be the point on the arc $(P, s)$ such that $l(w, s) = l(s, v_a)$, see Figure 4. We may assume without loss of generality that $w$ coincides with a processor node. Otherwise we can replace $w$ by dummy processor node at which no requests occur. If $e_j = \{v_j, v_{j+1}\}$ is the edge $w$ originally belonged to, then we can split $e_j$ into two edges $e_j^1 = \{v_j, w\}$ and $e_j^2 = \{w, v_{j+1}\}$. Since $a_\sigma(e_j^1) = a_\sigma(e_j^2) = a_\sigma(e_j)$, the total cost incurred by the two new edges equals the cost incurred by $e_j$, i.e.

$$c \cdot \min\{a_\sigma(e_j^1), r\} l(e_j^1) + c \cdot \min\{a_\sigma(e_j^2), r\} l(e_j^2) = c \cdot \min\{a_\sigma(e_j), r\} l(e_j).$$

Hence the introduction of a dummy processor node at $w$ does not change the cost $C_R(\sigma)$. In the remainder of this proof we will assume that the total number of processor nodes in the ring (including a possible dummy node) equals $n$. The nodes and edges are numbered in the way described in the beginning of this section.

Let

$$I_1 = \{i \in I | e_i \text{ belongs to the arc } (w, v_a)\}$$

and

$$I_2 = \{i \in I | e_i \text{ belongs to the arc } (v_a, w)\}.$$

We have

$$\begin{aligned} C_R(\sigma) &= c \sum_{i \in I} \min\{a_\sigma(e_i), r\} l(e_i) \\ &= c \sum_{i \in I_1} \min\{a_\sigma(e_i), r\} l(e_i) + c \sum_{i \in I_2} \min\{a_\sigma(e_i), r\} l(e_i) \\ &\leq c \sum_{i \in I_1} r l(e_i) + c \sum_{i \in I_2} a_\sigma(e_i) l(e_i). \end{aligned}$$

Furthermore,

$$\sum_{i \in I_1} l(e_i) = 2 \cdot l(s, v_a) \leq 2(l(s, v_a) + l(v_b, s))$$

and hence

$$\begin{aligned} C_R(\sigma) &\leq 2c(r l(s, v_a) + r l(v_b, s)) + c \sum_{i \in I_2} a_\sigma(e_i) l(e_i) \\ &\leq 2c(r l(s, v_a) + r l(v_b, s) + \sum_{i \in I_2} a_\sigma(e_i) l(e_i)). \end{aligned}$$

In the following we show

$$\sum_{i \in I_2} a_\sigma(e_i) l(e_i) \leq \sum_{j=1}^{k} \min\{l(v_a, v_{\mu(t_j)}), l(v_{\mu(t_j)}, v_b)\}. \tag{7}$$

Note that a request $\sigma(t)$ can only cause an access at an edge $e_i$, $i \in I_2$, if $v_{\mu(t)}$ belongs to $(v_a, w)$ and $v_{\mu(t)} \neq v_a$, $v_{\mu(t)} \neq w$. Let $\sigma(t'_1), \sigma(t'_2), \ldots, \sigma(t'_l)$ be all the requests in $\sigma$ satisfying these properties. Then we have

$$\begin{aligned} \sum_{i \in I_2} a_\sigma(e_i) l(e_i) &= \sum_{i \in I_2} \sum_{j=1}^{l} a_\sigma(e_i, t'_j) l(e_i) \\ &= \sum_{j=1}^{l} \sum_{i \in I_2} a_\sigma(e_i, t'_j) l(e_i) \\ &= \sum_{j=1}^{l} ( \sum_{\substack{i \in I_2 \\ i < \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) + \sum_{\substack{i \in I_2 \\ i \geq \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i)). \end{aligned}$$

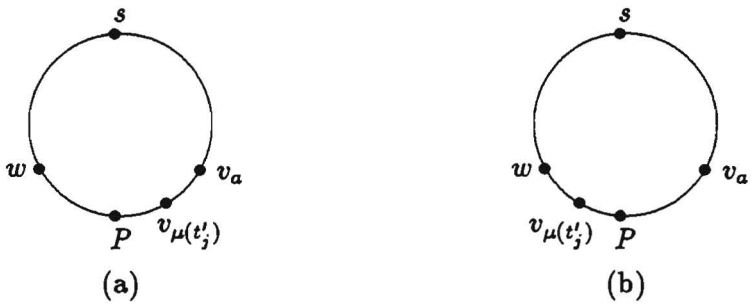



Figure 5: A request at $v_{\mu(t'_j)}$

Consider a fixed $j \in \{1, 2, \ldots, l\}$. If $v_{\mu(t'_j)}$ belongs to $(s, P)$, then $a_\sigma(e_i, t'_j) = 0$ for all $i \in I_2$ with $i \geq \mu(t'_j)$, see Figure 5(a). Thus

$$\begin{aligned}
\sum_{\substack{i \in I_2 \\ i < \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) + \sum_{\substack{i \in I_2 \\ i \geq \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) &= \sum_{\substack{i \in I_2 \\ i < \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) \\
&= \sum_{\substack{i \in I_2 \\ i < \mu(t'_j)}} l(e_i) \\
&= l(v_a, v_{\mu(t'_j)}) \\
&\leq \min\{l(v_a, v_{\mu(t'_j)}), l(v_{\mu(t'_j)}, w)\}.
\end{aligned}$$

If $v_{\mu(t'_j)}$ belongs to $(P, s)$ and $v_{\mu(t'_j)} \neq P$, then $a_\sigma(e_i, t'_j) = 0$ for all $i \in I_2$ with $i < \mu(t'_j)$, see Figure 5(b). Hence

$$\begin{aligned}
\sum_{\substack{i \in I_2 \\ i < \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) + \sum_{\substack{i \in I_2 \\ i \geq \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) &= \sum_{\substack{i \in I_2 \\ i \geq \mu(t'_j)}} a_\sigma(e_i, t'_j) l(e_i) \\
&= \sum_{\substack{i \in I_2 \\ i \geq \mu(t'_j)}} l(e_i) \\
&= l(v_{\mu(t'_j)}, w) \\
&\leq \min\{l(v_a, v_{\mu(t'_j)}), l(v_{\mu(t'_j)}, w)\}.
\end{aligned}$$

We obtain

$$\begin{aligned}
C_R(\sigma) &\leq 2c(rl(s, v_a) + rl(v_b, s) + \sum_{j=1}^{l} \min\{l(v_a, v_{\mu(t'_j)}), l(v_{\mu(t'_j)}, w)\}) \\
&\leq 2c(rl(s, v_a) + rl(v_b, s) + \sum_{j=1}^{l} \min\{l(v_a, v_{\mu(t'_j)}), l(v_{\mu(t'_j)}, v_b)\}).
\end{aligned}$$

Since $\sigma(t'_1), \sigma(t'_2), \ldots, \sigma(t'_l)$ is a subsequence of $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$ we have

$$\begin{aligned}
C_R(\sigma) &\leq 2c(rl(s, v_a) + rl(v_b, s) + \sum_{j=1}^{k} \min\{l(v_a, v_{\mu(t_j)}), l(v_{\mu(t_j)}, v_b)\} \\
&= 2c \cdot C_{OPT}(\sigma)
\end{aligned}$$

and inequality (6) is proved. □

**Proof of Theorem 4:** We may assume without loss of generality that the circumfence of the ring is 1. Let $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ be a request sequence for the ring. As in the proof of Theorem 3, let $v_a$ and $v_b$ be the nodes farthest from $s$ to which OPT replicates the page in clockwise and counter-clockwise direction, respectively.

Let $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_l)$ be the requests in $\sigma$ which request a node that does not belong to $(s, v_a)$ or $(v_b, s)$. For $k = 1, 2, \ldots, l$, let $v_{\mu(t_k)}$ be the node requested by $\sigma(t_k)$. The cost incurred by OPT in serving $\sigma$ equals

$$C_{OPT}(\sigma) = rl(s, v_a) + rl(v_b, s) + \sum_{k=1}^{l} \min\{l(v_a, v_{\mu(t_k)}), l(v_{\mu(t_k)}, v_b)\}.$$

In the following we show that the expected cost incurred by RING(RANDOM), $E[C_{RR}(\sigma)]$, satisfies

$$E[C_{RR}(\sigma)] \leq 2c \cdot C_{OPT}(\sigma). \tag{8}$$

This implies the theorem.

For the analysis of $E[C_{RR}(\sigma)]$ we need some more notation. Let $I = \{1, 2, \ldots, n\}$ and

$$I_1 = \{i \in I | e_i \text{ belong to the arc } (v_b, v_a)\}.$$

Set $I_2 = I \setminus I_1$. For $i = 1, 2, \ldots, n$, let $T_i$ be the tree that is obtained if the random cutting point $P$ is located on edge $e_i = \{v_i, v_{i+1}\}$ but $P \neq v_{i+1}$. Consider an arbitrary tree $T_i$, $1 \leq i \leq n$. Let $j \in \{1, 2, \ldots, n\}$ and $t \in \{1, 2, \ldots, m\}$ be arbitrary. We denote by $v_{\mu(t)}$ the node that is requested at time $t$. We set $a_\sigma^i(e_j, t) = 1$ if in the tree $T_i$, $e_j$ is on the path from $v_{\mu(t)}$ to the root. Otherwise we set $a_\sigma^i(e_j, t) = 0$. If $a_\sigma(e_j, t) = 1$, we say that $\sigma(t)$ *causes an access at the edge* $e_j$ *in the tree* $T_i$. We set $a_\sigma^i(e_j) = \sum_{t=1}^{m} a_\sigma^i(e_j, t)$. If $P$ is located on the edge $e_i = \{v_i, v_{i+1}\}$ but $P \neq v_{i+1}$, then by inequality (5) $RING(RANDOM)$ incurs a total cost of

$$\sum_{j=1}^{n} c \min\{a_\sigma^i(e_j), r\} l(e_j).$$

Since the circumfence of the circle is 1, with probability $l(e_i)$, $P$ is located on edge $e_i = \{v_i, v_{i+1}\}$ but $P \neq v_{i+1}$. Thus

$$E[C_{RR}(\sigma)] = \sum_{i=1}^{n} l(e_i) \sum_{j=1}^{n} c \min\{a_\sigma^i(e_j), r\} l(e_j).$$

Taking into account that $\sum_{j=1}^{n} l(e_j) = 1$ and $\sum_{j \in I_1} l(e_j) \leq 1$, we can show by algebraic manipulations of the above formula

$$\begin{aligned} E[C_{RR}(\sigma)] &\leq 2cr \sum_{i \in I_1} l(e_i) + c \sum_{i \in I_2} l(e_i) \sum_{j \in I_2} a_\sigma^i(e_j) l(e_j) \\ &= 2cr(l(s, v_a) + l(v_b, s)) + c \sum_{i \in I_2} l(e_i) \sum_{j \in I_2} a_\sigma^i(e_j) l(e_j). \end{aligned}$$

We show that

$$\sum_{i \in I_2} l(e_i) \sum_{j \in I_2} a_\sigma^i(e_j) l(e_j) \leq 2 \sum_{k=1}^{l} \min\{l(v_a, v_{\mu(t_k)}), l(v_{\mu(t_k)}, v_b)\}.$$

This proves inequality (8).

Consider an edge $e_j$ with $j \in I_2$. Note that if the randomly chosen cutting point $P$ is located on the arc $(v_a, v_b)$ and $P \neq v_b$, then only the requests $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_l)$ can cause an access at edge $e_j$. Hence

$$\begin{aligned}\sum_{i\in I_2} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j) l(e_j) &= \sum_{i\in I_2} l(e_i) \sum_{j\in I_2} \sum_{k=1}^{l} a^i_\sigma(e_j, t_k) l(e_j) \\ &= \sum_{k=1}^{l} \Big( \sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j, t_k) l(e_j) + \sum_{\substack{i\in I_2 \\ i\geq\mu(t_k)}} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j, t_k) l(e_j) \Big).\end{aligned}$$

Consider a fixed $k \in \{1, 2, \ldots, l\}$. If $P$ is located on the arc $(v_a, v_{\mu(t_k)})$ and $P \neq v_{\mu(t_k)}$, then $\sigma(t_k)$ cannot cause an access at an edge $e_j$ with $j < \mu(t_k)$. Thus $a^i_\sigma(e_j, t_k) = 0$ if $i < \mu(t_k)$ and $j < \mu(t_k)$. Hence

$$\sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j, t_k) l(e_j) = \sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) \sum_{\substack{j\in I_2 \\ j\geq\mu(t_k)}} a^i_\sigma(e_j, t_k) l(e_j) \leq \sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) \sum_{\substack{j\in I_2 \\ j\geq\mu(t_k)}} l(e_j).$$

Similarly we can show

$$\sum_{\substack{i\in I_2 \\ i\geq\mu(t_k)}} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j, t_k) l(e_j) = \sum_{\substack{i\in I_2 \\ i\geq\mu(t_k)}} l(e_i) \sum_{\substack{j\in I_2 \\ j<\mu(t_k)}} a^i_\sigma(e_j, t_k) l(e_j) \leq \sum_{\substack{i\in I_2 \\ i\geq\mu(t_k)}} l(e_i) \sum_{\substack{j\in I_2 \\ j<\mu(t_k)}} l(e_j).$$

We obtain

$$\sum_{i\in I_2} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j) l(e_j) \leq 2 \sum_{k=1}^{l} \sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) \sum_{\substack{j\in I_2 \\ j\geq\mu(t_k)}} l(e_j) \leq 2 \sum_{k=1}^{l} \min\{ \sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i), \sum_{\substack{j\in I_2 \\ j\geq\mu(t_k)}} l(e_j) \}$$

The last inequality follows because

$$\sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) \leq 1 \qquad \text{and} \qquad \sum_{\substack{j\in I_2 \\ j\geq\mu(t_k)}} l(e_j) \leq 1.$$

Since

$$\sum_{\substack{i\in I_2 \\ i<\mu(t_k)}} l(e_i) = l(v_a, v_{\mu(t_k)}) \qquad \text{and} \qquad \sum_{\substack{j\in I_2 \\ j\geq\mu(t_k)}} l(e_j) = l(v_{\mu(t_k)}, v_b).$$

we conclude

$$\sum_{i\in I_2} l(e_i) \sum_{j\in I_2} a^i_\sigma(e_j) l(e_j) \leq 2 \sum_{k=1}^{l} \min\{ l(v_a, v_{\mu(t_k)}), l(v_{\mu(t_k)}, v_b) \}$$

and inequality (8) is proved. □

## Acknowledgment

Useful discussions with Rudolf Fleischer are gratefully acknowledged.

## References


[ABF93] B. Awerbuch, Y. Bartal and A. Fiat. Competitive distributed file allocation. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 164-173, 1993.

[BFR92] Y. Bartal, A. Fiat and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. 24th Annual ACM Symposium on Theory of Computing*, pages 39-50, 1992.

[BBKTW94] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, special issue on on-line algorithms, **11**(1):2-14, 1994.

[BS89] D.L. Black and D.D. Sleator. Competitive algorithms for replication and migration problems. Technical Report Carnegie Mellon University, CMU-CS-89-201, 1989.

[CLRW93] M. Chrobak, L.L. Larmore, N. Reingold and J. Westbrook. Page migration algorithms using work functions. In *Proc. 4th International Annual Symposium on Algorithms and Complexity*, Springer Lecture Notes in Computer Science, Vol. 762 pages 406-415, 1993.

[DF82] D. Downey and D. Foster. Comparative models of the file assignment problem. *Computing Surveys*, **14**(2):287-313, 1982.

[K93] H. Koga. Randomized on-line algorithms for the page replication problem. In *Proc. 4th International Annual Symposium on Algorithms and Complexity*, Springer Lecture Notes in Computer Science, Vol. 762, pages 436-445, 1993.

[RS89] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. In *Proc. 16th International Colloquium on Automata, Languages and Progamming*, Springer Lecture Notes in Computer Science, Vol. 372, pages 687-703, 1989.

[SD89] C. Scheurich and M. Dubois. Dynamic page migration in multiprocessors with distributed global memory. *IEEE Transactions on Computers*, **38**(8):1154-1163, 1989.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, **28**:202-208, 1985.

[W92] J. Westbrook. Randomized Algorithms for the multiprocessor page migration. In *Proc. of the DIMACS Workshop on On-Line Algorithms*, American Mathematical Society, pages 135-149, 1992.