DIMACS Technical Report 98-36
September 1998

Broadcast Disk Paging with a Small Cache

by

Vincenzo Liberatore[1,2]

[1] Affiliated Graduate Student Member
[2] Department of Computer Science, Hill Center, Rutgers University, New Brunswick, NJ 08903. E-mail: `liberato@cs.rutgers.edu`. URL: `http://www.cs.rutgers.edu/~liberato/`. Supported by NSF Career Development Award CCR-9501942, NATO Grant CRG 960215, NSF/NIH Grant BIR 94-12594-03-CONF and an Alfred P. Sloan Research Fellowship. Travel support provided by a DIMACS graduate fellowhip.

ABSTRACT

We consider the broadcast disk paging problem when the cache size is 2 and n pages are transmitted. We show an algorithm that is $(n - 2/(n-2))$-competitive and prove that no better competitive ratio is possible.

1 Introduction

We have studied broadcast disk paging in [2]. Most of our proofs hold when the cache size k is at least 3 or some larger constant. In this paper, we will study the case $k = 2$, and we will show an algorithm that is $(n - 2/(n-2))$-competitive and prove that no better algorithm is possible. In §2, we define the broadcast disk paging problem. In §3, we prove that no on-line algorithm can be better than $(n - 2/(n-2))$-competitive. In §4, we define an algorithm called *adaptive-greedy* and show that it is $(n - 2/(n-2))$-competitive.

2 Preliminaries

In this section, we will establish our notation and describe basic properties of broadcast disks.

2.1 Notation

In a broadcast disk, a set $P = \{0, 1, \ldots, n-1\}$ pages are transmitted over a network in a round-robin fashion. The pages are received by the clients and at most k pages can be cached by any one client. A sequence of page requests $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_m \rangle \in P^m$ of length m is generated by applications running on the client. For notational convenience, let $\sigma_0 = k - 1$. If the requested page is in the client cache, then the application immediately obtains the requested page and continues without further ado. However, if the requested page is not locally available, the client has to wait for the page to be transmitted again over the network. Paging over a broadcast disk extends the virtual memory, except that now paging is from the network rather than from a physical disk. A page replacement algorithm affects the performance of the system. Disk paging and broadcast disk paging differ in the following essential point: paging replacement algorithms aim at minimizing the number of page faults whereas broadcast disk paging replacement algorithms aim at minimizing the total time waited for pages to arrive from the network.

If $p \in P$ is a page an i a non-negative integer, then we will write $p + i$ instead of $(p + i) \bmod n$ when no confusion can arise. If $p, r \in P$ are pages then we will write $p - r$ instead of $(p - r) \bmod n$ when no confusion can arise. Let G be a broadcast disk page replacement algorithm. The *configuration* of G is a pair $(M, i) \in 2^P \times P$, where M is a subset of P of size at most k of pages in G's local cache and i denotes the last page that was broadcast and received by G. If G's configuration is (M, i), we will say that G is *positioned* over page i. We will say that G moves if it changes the page where it is positioned. The algorithm G spends $p - r$ time units if it was positioned on r and moves to p. The total waiting time or cost of G on σ is denoted by $c(G, \sigma)$. We will say that G executes i rotations if $\lceil \frac{c(G,\sigma)}{n} \rceil \geq i$. If $M = \{p_1, p_2, \ldots, p_h\}$, then we will denote the memory configuration (M, p_1) as $\{\triangleright(p_1), p_2, \ldots, p_h\}$. The memory configuration reached by G before the jth request will be denoted as $\mathcal{M}(G, j)$ or simply as $\mathcal{M}(G)$ when the time index is clear from the context. We will assume without loss of generality that the initial configuration of any algorithm

G is $\mathcal{M}(G,1) = \{0, 1, \ldots, \triangleright(k-1)\}$. If $\mathcal{M}(G) = (M, p)$ then we will write $q \in \mathcal{M}(G)$ if and only if $q \in M$. If the jth request $\sigma_j \notin \mathcal{M}(G, j)$, then we will say that G *misses* or *faults*. If G and H are two algorithms, then we will write $\mathcal{C}(G, H, j)$ or simply $\mathcal{C}(G, H)$ for $(\mathcal{M}(G, j), \mathcal{M}(H, j))$ and we will say that $\mathcal{C}(G, H, j)$ is the *joint configuration* of G and H at step j.

3 Lower Bound

Theorem 3.1 *No deterministic algorithm for broadcast disk paging and $k = 2$ cache size can achieve a competitive ratio better than $n - \frac{2}{n-2}$.*

Proof. Let G be a deterministic online algorithm and H the algorithm that the adversary uses to satisfy the request sequence. We will assume that $n \geq 4$. The adversary proceeds into phases. At the beginning of the each phase, we will maintain that $\mathcal{M}(G) = \{\triangleright(p), q\}$ and $\mathcal{M}(H) = \{p, \triangleright(r)\}$. It follows that $p \neq r$. We will also maintain that at the beginning of each phase $r + 1 \notin \mathcal{M}(H)$.

Let $\gamma = n - \frac{2}{n-2}$ and define the potential function

$$\Phi(\mathcal{M}(H)) = (n - 2 - (p - r)) \left(\frac{n}{\gamma} - 1\right) .$$

In particular we note that when $r = p + 2$ the potential is 0, and that when the quantity $p - r$ decreases by 1, the potential increases by $\frac{n}{\gamma} - 1$.

Let h be the real cost of H during the current phase, $a = h + \Delta\Phi$ the amortized cost of H, and g the real cost of G. It is enough to show that $\gamma a \leq g$ in each phase.

In a few cases, we will have in the middle of a phase that $\mathcal{M}(G) = \mathcal{M}(H) = \{p, \triangleright(r)\}$ and $p \neq r + 1$. At this point, we cannot declare the phase to be over, and we will proceed as follows. The adversary requests $r + 1$. The next request depends on the behavior of G:

- If $\mathcal{M}(G) = \{r, \triangleright(r+1)\}$, the adversary moves to $\mathcal{M}(H) = \{p, \triangleright(r+1)\}$ for a unit cost and requests p. At this point the phase ends. In this case, H has paid a unit cost and g has paid $p - r$.

- If $\mathcal{M}(G) = \{p, \triangleright(r+1)\}$, the adversary moves to $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$ for a unit cost and requests r. At this point the phase ends. In this case, H has paid a unit cost and g has paid n.

The phase analysis is divided into cases. The first case is when $r + 1 \notin \mathcal{M}(G)$ and $r + 2 \notin \mathcal{M}(H)$. The adversary requests $r + 1$ and pays a unit cost. We have two subcases depending on whether G loads r while passing over it to reach $r + 1$:

- If G loads r, the configuration reached by G is $\{r, \triangleright(r+1)\}$. To satisfy $r + 1$, the adversary evicts r and then requests p. Now, G has to move back to p incurring a total phase cost of n. Moreover, $\mathcal{M}(G) = \{\triangleright(p), q'\}$ for some $q' \in P$, $\mathcal{M}(H) = \{p, \triangleright(r+1)\}$, and another phase starts. The costs are $g = n$, $a = 1$, $\Delta\Phi = \frac{n}{\gamma} - 1$, and

$$\gamma a \leq \gamma \left(1 + \frac{n}{\gamma} - 1\right) = n = g \ . \tag{1}$$

- If G does not load r, the configuration reached by G is $\{p', \triangleright(r+1)\}$ for some $p' \in P - \{r\}$. To satisfy $r+1$, the adversary evicts p and then requests r. Now, G has to move back to r incurring a total phase cost greater than n. Moreover, $\mathcal{M}(G) = \{\triangleright(r), q'\}$ for some $q' \in P$ and $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$ and another phase starts. The cost analysis is similar to that in the previous case, and, indeed, is even less favorable to the online algorithm G.

The second case is when $r+1 \in \mathcal{M}(G) - \mathcal{M}(H)$ and $r+2 \notin \mathcal{M}(H)$. Then, $p \neq r+1$, so that $r+1 = q$, and, by $p \neq r$, we obtain $r \notin \mathcal{M}(G)$, and the adversary requests r. The next requests of the adversary depend on the behavior of G.

- If G evicts p, the adversary requests p, and so $\mathcal{M}(H) = \{p, \triangleright(r)\}$, $\mathcal{M}(G) = \{\triangleright(p), q'\}$, $q' \in P$, and the phase is over. During the phase, the adversary had no amortized cost, while the cost of G was n.

- If $\mathcal{M}(G) = \{p, \triangleright(r)\}$, the two configurations are identical. Therefore, $h = 1$, $\Delta\Phi \leq \frac{n}{\gamma} - 1$, $g \geq (r-p) + (p-r) = n$, and the cost analysis is given by (1).

The third case is when $r+1 \notin \mathcal{M}(G)$ and $r+2 \in \mathcal{M}(H)$. Therefore, $p = r+2$. The adversary requests $r+1$ and pays a unit cost. Then,

- If G loads r, the configuration reached by G and by H is $\{r, \triangleright(r+1)\}$. We repeat the argument for the case of identical configurations. Eventually, $h = 2$, $\Delta\Phi = -(n-4)\left(\frac{n}{\gamma} - 1\right)$, and $g = 2(n-1)$. Then,

$$\gamma a = 2\gamma - (n-4)(n-\gamma) = 2n - \frac{4}{n-2} - (n-4)\frac{2}{n-2} = 2n - 2 = g \ . \tag{2}$$

- If G does not load r, the adversary requests it and the phase is over. Then $h = 1$, $\Delta\Phi \leq 0$, $g \geq n$.

The final case is when $r+1 \in \mathcal{M}(G)$ and $r+2 \in \mathcal{M}(H)$. Again, $p = r+2$ and $q = r+1$ so that $r \notin \mathcal{M}(G)$. The adversary requests r and G is charged a cost of $n-2$ while H has no cost. If G has evicted $r+2$, then $r+2$ it is requested for $g = n$, $h = 0$ and the phase ends. If G has not evicted $r+2$, then $\mathcal{M}(G) = \{\triangleright(r), r+2\}$. Now, the adversary requests $r+1$ and H moves to $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$. Correspondingly,

- If G evicts r, the adversary requests r for an additional cost of $n-1$ for G. The phase ends with $h = 1$, $g = 2n-3$, $\Delta\Phi < 0$ and $\gamma a < g$.

- If G ends up in $\mathcal{M}(G) = \mathcal{M}(H)$, we repeat the analysis for the case of identical configurations. Then, $g = 2(n-1)$, $h = 2$, $\Delta\Phi \leq -(n-4)\left(\frac{n}{\gamma} - 1\right)$, and (2) applies.

□

4 Adaptive-Greedy

We define now the online algorithm *adaptive-greedy* ($\mathcal{A}$). The marking algorithm achieves a competitive ratio even if it considers only one side of the trade-off, which is the minimization of the page faults. Another algorithm might try to minimize only the second performance measure, that is, the cost of the pages that are not loaded. Such an algorithm would work as follows: if σ_j causes a page fault, then the algorithm loads $\sigma_j, \sigma_j - 1, \ldots, \sigma_j - (k-1)$. Unfortunately, that algorithm is not competitive for $n = \Omega(k^2)$. Consider an adversary that loads and repeatedly requests pages $0, k, 2k, \ldots, (k-1)k$. Then, the online algorithm would fault at every request while the adversary would not pay any cost. The problem of this algorithm is lack of adaptiveness to the configuration of the adversary. So, the problem is to design an online algorithm that is adaptive to the configuration of the adversary like the marking algorithm is and that at the same time keeps into account the different page fault costs. $\mathcal{A}$ is such an algorithm and it is described below.

First of all, $\mathcal{A}$ is lazy. Therefore, $\mathcal{A}$ is always positioned on the last page that caused a fault. Moreover, $\mathcal{A}$ keeps track of the last evicted page, breaking ties in favor of the most recently used page, and we will denote that page as $p(\mathcal{A})$. In response to a request for page r, if $r \in \mathcal{M}(\mathcal{A})$, nothing happens. If the request causes a fault, then $\mathcal{A}$ proceeds as follows: if $r = p(\mathcal{A})$, then $\mathcal{A}$ moves to r and evicts the page $\arg\min\{p - r : p \in \mathcal{M}(\mathcal{A})\}$ — note that the evicted page is the one that is cheapest to reload once $\mathcal{A}$ is positioned on r. If $r \neq p(\mathcal{A})$, then $\mathcal{A}$ loads r and $r-1$. We will assume that initially $\mathcal{M}(\mathcal{A}, 1) = \{0, \triangleright(1)\}$. We will prove that $\mathcal{A}$ is strongly competitive. For the sake of clarity, we will first prove that $\mathcal{A}$ is n-competitive, and then that it is $(n - 2/(n-2))$-competitive, which is the best possible.

Lemma 4.1 *If $\mathcal{A}$ holds two consecutive pages, then it is positioned on the second one.*

Proof. If $\mathcal{A}$ satisfied the last faulting request σ_j with $\sigma_j \neq p(\mathcal{A})$, then the claim is trivially true. Suppose now that the last faulting request $\sigma_j = p(\mathcal{A})$. Then $\mathcal{A}$ will be positioned on σ_j. Furthermore, we claim that $\sigma_j + 1 \notin \mathcal{M}(\mathcal{A})$. Indeed, if $\sigma_j + 1$ was not in $\mathcal{A}$'s fast memory before the request to r, then $\mathcal{A}$ is lazy and does not load it. On the other hand, if $\sigma_j + 1$ was in $\mathcal{A}$'s fast memory before the request to σ_j, then $\sigma_j + 1$ is evicted by the rule for $\sigma_j = p(\mathcal{A})$.

□

Theorem 4.2 *The algorithm $\mathcal{A}$ is n-competitive when $k = 2$.*

Proof. Let H be the algorithm used by the adversary and assume without loss of generality that H is lazy. Assume also that the request sequence σ is hard for $\mathcal{A}$. Define a *phase* as a maximal subsequence of the request sequence σ with the property that H pays a cost only on the first request in a phase. We will show that in each phase the cost a of $\mathcal{A}$ is at most n times the cost h of H in that phase.

Suppose that the current phase begins with the jth request. $\mathcal{A}$ is positioned on σ_{j-1} at step j and $\sigma_{j-1} \neq \sigma_j$ because the sequence is hard. Moreover, a lazy adversary will be positioned on σ_j throughout the phase. Consider first the case when $h = 1$ and suppose that $\sigma_j \neq p(\mathcal{A})$. Then, $\mathcal{A}$ loads σ_j and $\sigma_j - 1$ and updates the value of $p(\mathcal{A})$. At this point either the phase ends and $a < n$, or the adversary requests σ_{j-1}. Therefore, $\mathcal{M}(H) = \{\sigma_{j-1}, \triangleright(\sigma_j)\}$. In this case, the new value of $p(\mathcal{A})$ is σ_{j-1} and $\mathcal{A}$ and H hold the same pages in fast memory, so the phase is over. Then $a = n \leq hn$.

Suppose now that $h = 1$ and $\sigma_j = p(\mathcal{A})$. First, we will show that $\sigma_{j-1} = \sigma_j - 1$. Suppose not. Since H is lazy and before step j is positioned on $\sigma_j - 1$, it had been positioned there since the last request to $\sigma_j - 1$. On the other hand, $\sigma_{j-1} \in \mathcal{M}(H, j)$, so that $\mathcal{M}(H, j-1) = \{\sigma_{j-1}, \triangleright(\sigma_j - 1)\}$. We claim that $\sigma_{j-2} = \sigma_j - 1$. Indeed, the sequence is hard for $\mathcal{A}$, and so $\sigma_{j-2} \neq \sigma_{j-1}$. Moreover, had the request been a $q \notin \{\sigma_{j-1}, \sigma_j - 1\}$, H could not have satisfied it, because it did not move. We conclude that the previous request was $\sigma_{j-2} = \sigma_j - 1$. Therefore, immediately before the request on σ_{j-1}, $\mathcal{A}$ was positioned on $\sigma_j - 1$, and since it evicts σ_j on the request to σ_{j-1}, we have $\mathcal{M}(\mathcal{A}, j-1) = \{\triangleright(\sigma_j - 1), \sigma_j\}$, which contradicts Lemma 4.1. We have then show that $\sigma_{j-1} = \sigma_j - 1$, and so $a = h = 1$.

Let us turn now to the case $h > 1$. Then, $a < 2n \leq hn$, which concludes the analysis of the algorithm. □

We prove that $\mathcal{A}$ is strongly competitive. First, we say that a joint configuration is a *race* if H is positioned on r, $p(\mathcal{A}) = r + 2$, and $r + 1 \notin \mathcal{M}(\mathcal{A})$. It is easy to see that if $\mathcal{C}(\mathcal{A}, H, j)$ is a race and $\sigma_j = r + 2$, then $r + 1 \notin \mathcal{M}(\mathcal{A}, j + 1)$. If $\mathcal{A}$ and H hold the same pages in fast memory, then we will say that the joint configuration of $\mathcal{A}$ and H are *concordant*, otherwise they are *discordant*. If the joint configuration is concordant, we will say that it is *bad* (for $\mathcal{A}$) if either $\mathcal{M}(H) = \{r-1, \triangleright(r)\}$ or $\mathcal{M}(H) = \{p, \triangleright(r)\}$ for $p \neq r+1$ and $\mathcal{M}(\mathcal{A}) = \{\triangleright(p), r\}$. A concordant configuration that is not bad for $\mathcal{A}$ will be said to be *good*.

We will say that the adversary *punishes* a discordant configuration if it repeatedly asks pages not in $\mathcal{M}(G)$ until the configuration is concordant. We will now show that an adversary cannot punish a discordant configuration indefinitely. Indeed, the adversary is not allowed to move from its position $\mathcal{M}(H) = \{p, r\}$, so that if $\mathcal{M}(\mathcal{A})$ starts with $\mathcal{M}(\mathcal{A}) = \{q, \triangleright(r)\}$, it first goes to $\{\triangleright(p), q\}$ with $p(\mathcal{A}) = r$, and assume without loss of generality that $q \neq r$, and at the next step $\mathcal{M}(\mathcal{A}) = \{p, \triangleright(r)\}$, and the joint configuration is now concordant.

We will also define a way for the adversary to punish a good concordant configurations. Let $\mathcal{M}(H) = \{p, \triangleright(r)\}$. If $p - r > 2$, then the adversary punishes a good concordant position by moving to $\{p, \triangleright(r + 1)\}$ and then punishing the resulting discordant configuration. If $p - r \leq 2$, then the adversary punishes a good concordant configuration by moving to $\{r, \triangleright(r + 1)\}$ and then punishing the resulting configuration.

Lemma 4.3 *For all $n > 3$, the configuration reached after the adversary punishes a good concordant configuration is not a race, is bad, and $\mathcal{A}$ does not pay more than $n-2$ while the adversary punishes the good configuration.*

Proof. The proof is by case analysis. In the case $p-r > 2$, $\mathcal{M}(\mathcal{A})$ will first be $\{r, \triangleright(r+1)\}$, then $\{r+1, \triangleright(p)\}$ and the cost of $\mathcal{A}$ is $p-r \leq n-2$. Since $p(\mathcal{A}) = r \neq r+3$ (for $n > 3$), the resulting configuration is not a race. In the case $p-r \leq 2$, $\mathcal{M}(\mathcal{A})$ will directly be $\{r, \triangleright(r+1)\}$ for no more than a unit cost. The final configuration would be a race if $p(\mathcal{A}) = p = r+3$, which contradicts $p - r \leq 2$. □

Theorem 4.4 *The algorithm $\mathcal{A}$ is strongly competitive.*

Proof. We need to prove that $\mathcal{A}$'s competitive ratio is equal to $\gamma = n - \frac{2}{n-2}$. The proof is based on a potential function. In the proof we will assume $n \geq 6$. The result is also proved for $n = 3, 4, 5$ by solving a linear program (see [1]).

First of all, we state the following special provision for race configurations: if the rules below yield a potential for a race configuration which is less than γ, then the potential of that race configuration is set to γ.

We turn now to define the potential function. The potential of a bad concordant configurations is defined as follows. If $\mathcal{M}(H) = \{r-1, \triangleright(r)\}$, then the potential is γ. Otherwise, if $\mathcal{M}(H) = \{p, \triangleright(r)\}$ for $p \neq r+1$, the potential is equal to $(n-1) + \frac{2}{n-2}((p-r)-2)$. In a bad configuration with $p \neq r-1, r+1$, $2 \leq p-r \leq n-2$, and so the maximum potential of a bad configuration is $\gamma + \frac{n-4}{n-2}$ and the minimum potential of a bad configuration is $n-1$.

The potential of a good concordant position is defined as follows. Let $\tilde{g}$ be the maximum cost paid by $\mathcal{A}$ when the adversary punishes the good concordant configuration and $\tilde{\Phi}$ the potential of the final bad concordant position. Then, the potential of the good concordant configuration is equal to $\tilde{g} + \tilde{\Phi} - \gamma$. We showed above that $\tilde{g} \leq n-2$, and the maximum potential of a good configuration is $n - 2 + \frac{n-4}{n-2}$. Let us turn now to evaluate the minimum potential of a good configuration. If $\tilde{\Phi} < \gamma$, then $\tilde{\Phi} - \gamma \geq -\frac{n-4}{n-2} > 1$ and H's configuration after the joint configuration has been punished is $\mathcal{M}(H) = \{p, \triangleright(r)\}$ with $p - r \geq 2$, so that $\tilde{g} \geq (p-r) + 1 \geq 3$, and the potential is more than 2. On the other hand, if $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$, then $\tilde{\Phi} = \gamma$, and if $\tilde{g} = 0$, then the potential is 0. The second smallest value of the potential of a good configuration is also for $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$ when also $\mathcal{A}$ moves by one time unit and it is equal to 1.

The potential of a discordant position is defined as the cost paid by $\mathcal{A}$ when the adversary punishes the discordant configuration plus the potential of the final concordant position.

In the rest of the proof, we will evaluate the amortized cost a_j of $\mathcal{A}$ on request j and compare it with the cost h_j of H. The value g_j will be the real cost of $\mathcal{A}$ on request j. We will assume without loss of generality that the sequence σ is hard and H is lazy. It follows that $\mathcal{A}$ is always positioned on a page belonging to $\mathcal{M}(H)$.

Suppose first that $h_j = 0$. Since the request sequence is hard, $\mathcal{C}(\mathcal{A}, H, j)$ is a discordant configuration. We examine now the circumstances that create a race for $h_j = 0$. Suppose that

H is positioned over r. The new value of $p(\mathcal{A})$ is $r+2$, then $r+2 \in \mathcal{M}(\mathcal{A}, j)$ and it is evicted in response to σ_j. Since $\mathcal{A}$ is positioned on a page belonging to $\mathcal{M}(H)$, if $r+2 \notin \mathcal{M}(H)$, then $\mathcal{A}$ is positioned over r at step j. Then, σ_j is the page in $\mathcal{M}(H) - \{r\}$. If $\sigma_j \neq r+1$, then $\mathcal{A}$ evicts r on σ_j and, since $\mathcal{A}$ is over r, the new value of $p(\mathcal{A}) = r$, so that the new configuration is not a race. Assume now $\sigma_j = r+1$. Then, $r+1 \in \mathcal{M}(\mathcal{A}, j+1)$, and the new configuration is not a race. On the other hand, if $r+2 \in \mathcal{M}(H)$, then $\sigma_j = r$, and, since $\mathcal{A}$ evicts $r+2$, it must be that case that at step j $p(\mathcal{A}) \neq r$ and so $r-1 \in \mathcal{M}(\mathcal{A}, j+1)$. The configuration is race and discordant, and the potential is $\max\{\gamma, 2+n-1\} = n+1$. Therefore, even if a race configuration $\mathcal{C}$ is created, $\Phi(\mathcal{C})$ is determined only by $\mathcal{C}$'s punishment. Since $\mathcal{C}(\mathcal{A}, H, j)$ is discordant and $h_j = 0$, the present request is exactly the first step taken by an adversary that punishes the discordant configuration. Therefore, the potential change is at least equal to $-g_j$, and the analysis of this case is complete.

Suppose now that $h_j = 1$. Suppose first that $\mathcal{C}(\mathcal{A}, H, j)$ is discordant. Then g_j can be bounded by the sum of the cost paid by $\mathcal{A}$ when the adversary punishes $\mathcal{C}(\mathcal{A}, H, j)$ plus the cost of processing the request from that configuration. The first part is compensated by the drop in potential, while the second will reduce to the case of a concordant configuration, which is analyzed below.

Suppose that $\mathcal{C}(\mathcal{A}, H, j)$ was good. First notice that $\mathcal{M}(\mathcal{A}, j+1) = \{r-1, \triangleright(r)\}$) and $g_j \leq 1$. We examine now the circumstances under which a race is created. If $\mathcal{M}(H, j+1) = \{r-1, \triangleright(r)\}$, then $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) = \gamma$ independently of $p(\mathcal{A})$. Otherwise, since $p(\mathcal{A}) = r+2 \in \mathcal{M}(\mathcal{A}, j)$ and $\mathcal{C}(\mathcal{A}, H, j)$ is good, then $r+2 \in \mathcal{M}(H, j)$, so that $\mathcal{M}(H, j) = \{\triangleright(r-1), r+2\}$. Therefore, $\Phi(\mathcal{C}(\mathcal{A}, H, j+1) \geq 2+n-1 = n+1 \geq \gamma$. In either case, the potential does not depend from the fact that $\mathcal{C}(\mathcal{A}, H, j+1)$ is a race. We now turn to evaluate $\Delta\Phi$. There are two ways $\mathcal{C}(\mathcal{A}, H, j)$ is punished. If $\mathcal{M}(\mathcal{A})$ immediately becomes equal to $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$, then, $\tilde{g} = 1$, $\tilde{\Phi} = \gamma$, so that $\Phi(\mathcal{C}(\mathcal{A}, H, j)) = 1$, and moreover $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) = \gamma$ so that $a_j = \gamma$. Otherwise, let $\tilde{g}_2 = \tilde{g} - 1$, and $\Phi(\mathcal{C}(\mathcal{A}, H, j)) = \tilde{g} + \tilde{\Phi} - \gamma$, $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) = \tilde{g}_2 + \tilde{\Phi}$, and so $a_j = \gamma$.

Suppose now that $\mathcal{C}(\mathcal{A}, H, j)$ was bad. If $\mathcal{M}(H, j) = \{r-1, \triangleright(r)\}$, then, by Lemma 4.1, $g_j = 1$, $\Delta\Phi \leq n-2+\frac{n-4}{n-2}$, so that $a_j = \gamma$. Otherwise, if $\mathcal{M}(H, j) = \{p, \triangleright(r)\}$ for $p \neq r-1, r+1$, then $g_j = 1+r-p$, and it is easy to see that $\mathcal{M}(\mathcal{A}, j+1) = \{r, \triangleright(r+1)\}$. Therefore, $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) = (p-r) - 1 + \Phi(\mathcal{C}(\mathcal{A}, H, j)) - \frac{2}{n-2}$, and so $\Delta\Phi \leq n - g_j - \frac{2}{n-2}$. The analysis of the case when a discordant configuration is reached is then concluded.

It remains to analyze the case when the cost of H is $h_j \geq 2$. Clearly, $g_j \leq n-1$. Since H is lazy, we can assume that after σ_j H and $\mathcal{A}$ are positioned over the same page.

Suppose first that $\mathcal{C}(\mathcal{A}, H, j+1)$ is concordant. Since H and $\mathcal{A}$ are positioned on the same page, then either the configuration is bad and $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) = \gamma$, or the configuration is good and $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) \leq n-2+\frac{n-4}{n-2} \leq \gamma$, and the amortized cost is no more than

$$n - 1 + \gamma \leq 2\gamma \leq h_j\gamma \ .$$

Suppose now that $\mathcal{C}(\mathcal{A}, H, j+1)$ is a discordant configuration. We now estimate $\Phi(\mathcal{C}(\mathcal{A}, H, j+1))$. If $\mathcal{C}(\mathcal{A}, H, j+1)$ is a race, then $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) \geq \gamma$. If $\mathcal{C}(\mathcal{A}, H, j+1)$ is not a race, then the potential is estimated by evaluating the costs incurred when the adversary punishes it. Suppose that a bad configuration is reached after the punishment. If $\mathcal{M}(H) = \{r, \triangleright(r+1)\}$ in that bad configuration, $\mathcal{A}$ has paid a cost of n during the punishment, and $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) = n + \gamma$. In all other bad configurations, $\mathcal{A}$ has not paid more than $n-2$ during the punishment and $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) \leq n - 2 + \gamma + \frac{n-4}{n-2} < n + \gamma$. If the adversary punishes $\mathcal{C}(\mathcal{A}, H, j+1)$ and a good configuration is reached, then $\mathcal{A}$ pays n, and $\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) \leq 2n - 2 + \frac{n-4}{n-2} \leq n + \gamma$. Therefore, the amortized cost is at most $n - 1 + n + \gamma = 3n - 1 - \frac{2}{n-2}$, which is less than 3γ for all $n \geq 6$.

We continue the analysis of the case when $h_j = 2$ and $\mathcal{C}(\mathcal{A}, H, j+1)$ discordant as follows: we divide this case into three subcases, depending on $\mathcal{C}(\mathcal{A}, H, j)$.

If $\mathcal{C}(\mathcal{A}, H, j)$ was a race, then $\Phi(\mathcal{C}(\mathcal{A}, H, j)) \geq \gamma$, and the amortized cost is no more than

$$n - 1 + n + \gamma - \gamma = 2n - 1 \leq 2\gamma = h_j\gamma \ ,$$

for all $n \geq 6$. Suppose now that $\mathcal{C}(\mathcal{A}, H, j)$ was not a race. Then, since $h_j = 2$, $\mathcal{M}(\mathcal{A}, j+1) = \{r, \triangleright(r+1)\}$.

If $\mathcal{C}(\mathcal{A}, H, j)$ was good, then actually $g_j \leq 2$ and we have two cases. If $\Phi(\mathcal{C}(\mathcal{A}, H, j)) = 0$, then in fact $g_j = 1$ and it is easy to check that the amortized cost is

$$n + \gamma - 1 + \frac{n-4}{n-2} = 2n - 1 + \frac{n-6}{n-2} < 2n - \frac{2}{n-2} = 2\gamma \ .$$

If $\Phi(\mathcal{C}(\mathcal{A}, H, j)) \neq 0$, then $\Phi(\mathcal{C}(\mathcal{A}, H, j)) \geq 1$. We now find the corresponding maximum value of $\Phi(\mathcal{C}(\mathcal{A}, H, j+1))$ by punishing $\mathcal{C}(\mathcal{A}, H, j+1)$. If $\mathcal{C}(\mathcal{A}, H, j+1)$ is punished and $\mathcal{M}(\mathcal{A})$ does not immediately become equal to $\mathcal{M}(H)$, then $\mathcal{A}$ pays no more than n and reaches a good configuration, so that

$$\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) \leq 2n - 2 + \frac{n-4}{n-2} \ .$$

If $\mathcal{C}(\mathcal{A}, H, j+1)$ is punished and $\mathcal{M}(\mathcal{A})$ immediately becomes equal to $\mathcal{M}(H)$, then $\mathcal{A}$ has not paid more than $n-2$ and

$$\Phi(\mathcal{C}(\mathcal{A}, H, j+1)) \leq n - 2 + \gamma + \frac{n-4}{n-2} \leq 2n - 2 + \frac{n-4}{n-2} \ .$$

Therefore the amortized cost is at most

$$2 + 2n - 2 + \frac{n-4}{n-2} - 1 = 2\gamma \ ,$$

which concludes the proof. □

References

[1] Sandy Irani and Anna R. Karlin. Online computation. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 13, pages 521–564. PWS Publishing Company, Boston, 1997.

[2] Sanjeev Khanna and Vincenzo Liberatore. On broadcast disk paging. In *Proceedings of the 30th ACM Symposium on the Theory of Computing*, pages 634–643, 1998.