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

[2] Swarup Acharya, Michael Franklin, and Stanley Zdonik. Prefetching from a broadcast disk. In *Proceedings of the International Conference on Data Engineering*, 1996.

[3] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. In *Algorithms—ESA '96 (Barcelona)*, volume 1136 of *Lecture Notes in Comput. Sci.*, pages 419–430, Berlin, 1996. Springer.

[4] L. A. Belady. A study of replacement algorithms for a virtual storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

[5] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[6] T. G. Bowen, G. Gopal, G. Herman, T. Hickey, K. C. Lee, W. H. Mansfield, J. Raitz, and A. Weinrib. The Datacycle architecture. *Communications of the ACM*, 35(12):71–81, December 1992.

[7] P. Cao, E. W. Felten, A. R. Karlin, and K. Li. A study of integrated prefetching and caching strategies. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 188–196, 1995.

[8] A. Fiat, R. Karp, M. Luby, L. A. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[9] Michael Franklin and Stanley Zdonik. A framework for scalable dissemination-based systems. In *Proceedings of the International Conference Object Oriented Programming Languages Systems*, pages 94–105, 1997.

[10] David K. Gifford. Polychannel systems for mass digital communications. *Communications of the ACM*, 33(2):141–151, February 1990.

[11] Tomasz Imielinski and B. Badrinath. Mobile wireless computing: Challenges in data management. *Communications of the ACM*, 37(10):18–28, October 1994.

[12] 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.

[13] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[14] Vincenzo Liberatore. Caching and scheduling for broadcast disk systems. Technical Report 98-71, UMIACS, 1998.

[15] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

[16] Lyle A. McGeoch and Daniel D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

[17] Efrem Sigel. *Videotext: The Coming Revolution in Home/Office Information Retrieval.* Knowledge Industry Publications, White Plains, 1980.

request. The algorithm G is free to decide whether page $i+1$ should be cached or not. Let G be an algorithm for BDP. The algorithm G can be turned into a BDP algorithm G^D in the delay model as follows. On a page request, G^D behaves exactly as the algorithm G would on that request. However, when G^D receives a delay request, it executes at most one full rotation after serving the delay request and returns to the configuration that it had before the delay.

Proposition 6.1 *If G is c-competitive for BDP against an ubiquitous adversary for some $c = \Omega(n)$, then G^D is $O(c)$-competitive in the delay model.*

Proof. Let σ^D be a request sequence consisting of page requests and delays, σ the request sequence where all delays have been removed, and d the number of delay requests in σ^D. Let f be the minimum number of faults on σ and H be the adversary algorithm. Observe that $c(H, \sigma^D) \geq f + d$. Then, $c(G^D, \sigma^D) = c(G, \sigma) + nd \leq cf + nd + b \leq O(c)(f + d) + b \leq O(c)c(H, \sigma^D) + b$, for some constant b. It is then proved that G is $O(c)$-competitive. $\square$

An immediate corollary of the above proposition is as follows.

Corollary 6.2 *There is an $O(n \log k)$-competitive randomized algorithm for the delay model with no prefetching and an $O(n \log k)$ deterministic algorithm for the delay model with prefetching.*

Proof. The first result follows from the $O(\log k)$-competitive lazy randomized algorithm for virtual memory paging [3, 8, 16] while the second result follows from the fact that the gray algorithm is $O(n \log k)$-competitive against an ubiquitous adversary. $\square$

7 Concluding Remarks

We studied deterministic as well as randomized algorithms for broadcast disk paging. An interesting question not resolved by our work is that of the competitive ratio of randomized prefetching algorithms. A lower bound of $\Omega(n)$ is easy to show. It is conceivable that a simultaneous use of randomization and prefetching yields an algorithm that is $o(n \log k)$-competitive.

Recently, the second author performed an empirical evaluation of the gray algorithm on both synthetic and Web traces and found that the gray algorithm always and consistently outperformed LRU [14].

Acknowledgments

We thank Martin Farach-Colton for many discussions, Uli Kremer and Stefan Langerman for telling us about the German and Belgian videotext systems, and two anonymous referees for helping us to improve the presentation of this paper.

References

[1] Swarup Acharya, Rafael Alonso, Michael Franklin, and Stanley Zdonik. Broadcast disks: Data management for asymmetric communication environments. In *Proceedings of the ACM SIGMOD Conference*, pages 199–210, 1995.

- by β_l, the number of pages that are gray at the lth request ($j \leq l \leq j+q$),
- by g_l, the number of gray pages requested from the beginning of the phase up to step $l-1$ inclusive,
- by w_l, the number of white pages requested from the beginning of the phase up to step $l-1$ inclusive, and
- by $b_l = w_l + g_l$, the number of black pages before the lth request.

The notation β_l is related to that in Lemma 5.2 by $\beta_l = \zeta_{l-1}$. Notice that $\beta_l = k - g_l$ and that $\alpha_l = \beta_l - q_l = k - g_l - (k - b_l) = g_l + w_l - g_l = w_l = w_j$ is a constant throughout the current gray block. Moreover, $\alpha_l \leq w_{k+1}$. Observe that w_{k+1} is the number of white requests in $\langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle$, w is the number of white requests in $\langle \sigma_2, \ldots, \sigma_k, \sigma_{k+1} \rangle$, and both σ_1 and σ_{k+1} are white requests. Therefore, $w_{k+1} = w$ and $\alpha_l \leq w$.

$\mathcal{M}(\mathcal{G}, l)$ contains all but at most w gray pages, and the missing gray pages are the closest to its current position σ_{l-1}. Thus, once $\mathcal{G}$ is positioned at σ_{l-1}, no more than w gray pages lie in the closed interval between σ_{l-1} and σ_l ($l = j+1, \ldots, j+q$). Hence, if $\mathcal{G}$ starts from σ_j and moves for more than n units, then there are at most $\beta_j - \left\lceil \frac{\beta_j+1}{w} \right\rceil \leq \beta_j \left(1 - \frac{1}{w}\right)$ pages that are still gray. Hence, if during the block B_i, $\mathcal{G}$ pays $r_i n + a$, for some $a < n$, then $\beta_{j+q+1} \leq (1 - 1/w)^{r_i} \beta_j$. On the other hand, since $\beta_2 = k$, if $\sum r_i > w \ln k$, no gray pages could be remaining since

$$\left(1 - \frac{1}{w}\right)^{\sum r_i} k < \left(\frac{1}{e}\right)^{\ln k} k = 1.$$

We conclude that $\sum r_i \leq w \ln k$. Notice that the quantity a contributes for at most $n-1$ per gray block to the cost on gray requests. Recall that σ_j contributed for another $n-1$ term per gray block. Since there are at most w gray blocks, the cost paid during all gray blocks is at most $2w(n-1) + n \sum r_i \leq w(2(n-1) + n \ln k)$.

5.8 White Requests

Henceforth, we will assume that $\mathcal{G}$ processes gray requests for free, but each white request is charged $2(n-1) + n \ln k$. Such charge is in addition to the cost required to process the white request itself. Notice that on any white request, $\mathcal{G}$ pays a real cost of at most $n-1$, is charged $2(n-1) + n \ln k$ for gray requests, and so $\mathcal{G}$'s amortized cost is $3(n-1) + n \ln k = \gamma$ plus any increase of the potential function. Suppose that σ_j is a white request that causes H to fault. The potential does not increase, and $\mathcal{G}$'s amortized cost is γ. Suppose now that σ_j is a white request that does not causes H to fault. Then, p is hidden, the potential decreases by γ, and $\mathcal{G}$'s amortized cost is naught. Finally, G pays a cost of γD at the end of the phase. On the whole, $\mathcal{G}$'s cost in the segment is $w^N \gamma + D\gamma$, which is no more than $2\gamma = O(n \log k)$ times the actual cost of H in the segment, and the proof is complete.

6 Delay Model

We will now describe the delay model for broadcast disk paging. In the delay model, the adversary has the power of issuing requests of two types: page requests, like in the ordinary BDP, and delays, where one delay request forces any algorithm to listen to one more page. Hence, if the algorithm G is positioned over page i before a delay request, G will be positioned over $i+1$ after the delay

5.5 Potential function

Let us turn now to evaluate the cost of $\mathcal{G}$ in the same segment. Let $\gamma = 3(n-1) + n \ln k$. We will denote by D_j, the number of hidden pages at step j, and so $D = D_{k+1}$. Define the potential function at step j to be

$$\Phi(j) = (3(n-1) + n \ln k) D_j = \gamma D_j \ ,$$

Clearly, $\Phi(j) \geq 0$ for all j's. We will analyze the amortized cost of $\mathcal{G}$ by considering the following cases:

- In the first case, $\mathcal{G}$ does not pay any real cost, but the potential might increase as a consequence of an increase of the number D_j of hidden pages.
- In the next two cases, $\mathcal{G}$ indeed pays a real cost.
    - We will examine the cost of $\mathcal{G}$ on gray requests, and finally
    - We will examine the cost of $\mathcal{G}$ on white requests.

5.6 Potential Increase

First, we show that the potential increases only at the end of a phase. We will need the following lemma.

Lemma 5.6 *If a page p is hidden at step j ($2 \leq j \leq k$), then p was hidden at step 2.*

Proof. It is enough to show that if a page p is hidden at step j ($2 < j \leq k+1$), then p was hidden at step $j-1$. Suppose to the contrary that p is hidden at step j, but not at step $j-1$. Then, either p is not white at step $j-1$ or $p \notin \mathcal{M}(H, j-1)$. However, $j \leq k$, so that no gray page has turned white. Hence, it must be the case that $p \notin \mathcal{M}(H, j-1)$. Since H does not prefetch, $p = \sigma_{j-1}$, and so p is black at step j, which is a contradiction, and the lemma is proven. □

Lemma 5.6 implies that $D_{j-1} \geq D_j$ for $j = 2, \ldots, k$. It follows that the potential increase increases only at the end of a phase when some gray pages become hidden white pages. The potential increase is γD.

5.7 Gray requests

We now evaluate $\mathcal{G}$'s cost on the gray requests. Define a *gray block* as a maximal sequence of gray requests followed by a white request. The segment can be partitioned into a sequence of gray blocks that alternate with white requests. Notice that there are at most w gray blocks in a phase, where w was defined as the number of white requests in the segment. Actually, the first phase request and the last segment request are both white, so that w is also the number of white requests in the phase. We will argue that the cost incurred by $\mathcal{G}$ during the gray blocks does not exceed $w(2(n-1) + n \ln k)$.

Consider a gray block $B_i = \langle \sigma_j, \sigma_{j+1}, \ldots, \sigma_{j+q} \rangle$. Since σ_{k+1} is a white request, $j + q \leq k$. $\mathcal{G}$'s cost for σ_j is at most $n-1$. We now estimate $\mathcal{G}$'s cost in the rest of the block. In what follows, we denote

- by α_l, the number of gray pages that are missing from $\mathcal{G}$'s fast memory before the lth request ($j \leq l \leq j+q$),

Definition 5.1 *A* segment *is a subsequence of the form*

$$\langle \sigma_{ik+2}, \sigma_{ik+3}, \ldots, \sigma_{(i+1)k+1} \rangle$$

for some $i \geq 0$.

The notion of segment will be central to the rest of the proof because segments allow us to compute algorithm cost. We will estimate the cost of $\mathcal{G}$ and H on each segment and prove that the cost of $\mathcal{G}$ in a segment is no more than $O(n \log k)$ the cost of H in the same segment.

From now on, we will fix our attention on the first segment for simplicity of notation and without loss of generality. The first segment consists of the request sequence $\langle \sigma_2, \sigma_3, \ldots, \sigma_{k+1} \rangle$. Recall that σ_{k+1} is a white request.

5.4 Cost of H

We turn now to examine the cost of H in a segment. Define w^N to be the number of white pages that are requested in the segment and that cause H to fault. Clearly, the cost of H is at least w^N. Suppose that on a white request for a page p, H already has p in its fast memory. Then the ratio of the actual costs of $\mathcal{G}$ and H for the request to p is infinity. This scenario thus requires a careful analysis as we describe below.

Definition 5.2 *A page p is* hidden *at step j if p is white before the jth request and $p \in \mathcal{M}(H, j)$.*

Broadly speaking, our objective is to show that H also implicitly pays a cost on the hidden white pages. We will prove that the cost of H on the segment is at least D, where D is the number of hidden pages at step $k+1$.

Lemma 5.4 *If a page p is hidden at step $k+1$, then p has not been requested in the segment.*

Proof. If $p \in \{\sigma_2, \ldots, \sigma_k\}$, then p is black after step k and becomes gray before step $k+1$. Therefore, p is not white at step $k+1$ and so p is not hidden at step $k+1$. □

Let p be a hidden page at step $k+1$. Notice that $p \in \mathcal{M}(H, k+1)$, p was not requested in the segment, and H has not prefetched p. We conclude that p has been in $\mathcal{M}(H)$ throughout the segment.

Lemma 5.5 *The cost of H in a segment is at least D.*

Proof. The proof is reminiscent of those in [12, 13, 18]. Clearly, $\sigma_1 \in \mathcal{M}(H, 2)$. Moreover, the previous lemma implies that all D pages hidden at step $k+1$ are in $\mathcal{M}(H, 2)$. Within a segment, the requests $\sigma_2, \ldots, \sigma_k$ are for pages that are not hidden at step $k+1$ and that are not σ_1. At most $k-D-1$ of those $k-1$ pages are in $\mathcal{M}(H, 2)$, and so the cost of H is at least $k-1-(k-D-1) = D$. □

On the whole, the cost of H in the segment is at least $\max\{w^N, D\} \geq (w^N + D)/2$.

$j+1$, and so it is not white at step j. It follows that p is gray at both steps. If p is in the interval from σ_j to σ_{j+1}, then it can be prefetched at no cost while $\mathcal{G}$ moves from σ_j to σ_{j+1}. We now show that no page that lies in the interval from σ_{j+1} to σ_j is prefetched by $\mathcal{G}$. Suppose, by contradiction, that such a page p is prefetched. First, we observe that σ_{j+1} does not start a new phase. Indeed, if σ_{j+1} started a new phase, then $\mathcal{M}(\mathcal{G}, j)$ is exactly the set of gray pages immediately before step $j+1$. However, p is gray and $p \notin \mathcal{M}(\mathcal{G}, j)$. We conclude that σ_{j+1} did not start a new phase. Therefore, $b_{j+1} = b_j + 1$ and so $q_{j+1} = q_j - 1$. Let ζ_j be the number of gray pages remaining after the jth request and notice that $\zeta_{j+1} \geq \zeta_j - 1$. Since $p \in \mathcal{M}(\mathcal{G}, j+1)$, p has one of the q_{j+1} largest values of $p - \sigma_{j+1}$ among all ζ_{j+1} pages that are gray at step $j+1$. Therefore, there are at least $\zeta_{j+1} - q_{j+1} \geq \zeta_j - q_j$ gray pages between σ_{j+1} and p. Since $p \notin \mathcal{M}(\mathcal{G}, j)$, p does not have one of the q_j largest values of $p - \sigma_j$ among all ζ_j gray pages at step j. Therefore, there are at most $\zeta = \zeta_j - q_j - 1$ gray pages between σ_j and p at step j. Since the number of gray pages never increases during a phase, there are at most ζ gray pages between σ_j and p at step $j+1$. However, $p - \sigma_{j+1} < p - \sigma_j$, which is to say that the interval from σ_{j+1} to p is contained in the interval from σ_j to p. Hence, there are at most $\zeta < \zeta_{j+1} - q_{j+1}$ gray pages between σ_{j+1} and p. Thus, we reach a contradiction and the lemma is proven. □

Remark. The gray algorithm is similar to a marking algorithm once we identify black pages with the marked pages and white pages with the unmarked pages. However, the gray algorithm differs from the marking algorithm in one important respect: an evicted page might be prefetched and reloaded later on in the phase, without being requested again. By Lemma 5.2, the prefetch operation is executed at no cost. Thus, the gray algorithm adjusts the eviction pattern dynamically according to the requests in a phase. We would like to point out here that an analysis of the standard marking algorithm explicitly uses three marks, and marks all n pages [3].

Theorem 5.3 *The gray algorithm $\mathcal{G}$ is $O(n \log k)$-competitive even against an ubquitous adversary.*

In the rest of the section, we will prove Theorem 5.3. Let H be the adversary's algorithm and assume without loss of generality that H is lazy. We will assume without loss of generality that the request sequence σ is hard for $\mathcal{G}$. No page is requested more than once in a phase and there are exactly k requests in a phase. Since all black pages are in $\mathcal{G}$'s fast memory, all requests are for gray and white pages. Correspondingly, we will say that a request is gray (white) if the requested page is gray (white) immediately before it is requested. The first request of a phase is white because all gray pages are in $\mathcal{G}$'s fast memory at the beginning of the phase.

The proof is structured as follows.

- We begin by defining the notion of a *segment*; segments allow us to give a lower bound on the cost of H.
- We proceed to examine $\mathcal{G}$'s cost on gray requests in a given segment, and show that it is at most $O(wn \log k)$, where w is the total number of white requests in the segment.
- Finally, we use a potential function argument to show that the amortized cost of $\mathcal{G}$ is no more than $O(n \log k)$ times the cost of H.

5.3 Segments

We now define the notion of segments. Segments start from the second request of a phase and end with the first request of the next phase.

step G has at least 2 pages missing from the set W and we always request the missing page that is farthest. Thus G must complete a rotation after $\lceil (x_i + 1)/2 \rceil$ requests.

Observe that $x_1 = k$ and that upon termination, $x_{t+1} = 2$, and so $2 \leq x_i \leq k$ for all $i = 1, 2, \ldots, t+1$. Furthermore, $x_{i+1} = x_i - |Z_i| \geq \lfloor (x_i - 1)/2 \rfloor \geq (x_i/2) - 1$. It is then straightforward to see that $t = \Omega(\log k)$. The cost of G during a phase is thus at least $nt = \Omega(n \log k)$.

Putting It All Together: Observe that the adversary can issue $(n-k)/2$ consecutive phases. Then, it will reload $\{0, 1, \ldots, \rhd(k-1)\}$ and another sequence of $(n-k)/2$ phases starts. The total cost of the adversary in such a sequence of phases is n while the total cost of G is $\Omega(n^2 \log k)$, and the result follows. □

5.2 An Upper Bound: The Gray Algorithm

We will now design a paging algorithm $\mathcal{G}$, referred to as the *gray algorithm*, that is $O(n \log k)$-competitive. The gray algorithm uses a set of three marks $\{\mathsf{black}, \mathsf{gray}, \mathsf{white}\}$ and maintains a mark for each page in P. Notice that $\mathcal{G}$'s marking policy is somewhat different from that of the marking algorithm [13] that uses only two marks, and marks only the pages in fast memory. We define b_j to be the number of black pages immediately before the jth request. The gray algorithm $\mathcal{G}$ works as follows:

- Initially, $\mathcal{G}$ marks the pages $\{0, 1, \ldots, k-1\}$ black and all other pages white.
- $\mathcal{G}$ ignores all requests that do not cause a fault. Henceforth, assume that σ is hard for $\mathcal{G}$.
- $\mathcal{G}$ works in phases:
  - a new phase is started when $b_j = k$.
  - at the beginning of a phase, $\mathcal{G}$ marks all gray pages white and all black pages gray.
- When $\mathcal{G}$ faults on σ_j, $\mathcal{G}$ loads σ_j and marks it black.
- Before the jth request, $\mathcal{G}$ keeps in its fast memory
  - the b_j black pages, plus
  - the set of $q_j = k - b_j$ gray pages $\gamma_1, \gamma_2, \ldots, \gamma_{q_j}$ that have the q_j smallest values of $\sigma_{j-1} - \gamma_i$ $(i = 1, 2, \ldots, q_j)$. In other words, the gray pages $\gamma_1, \gamma_2, \ldots, \gamma_{q_j}$ are the q_j gray pages that are most expensive to reload from σ_{j-1}.

We have stated the gray algorithm by assuming that a mark is associated with all n pages. In fact, $\mathcal{G}$ needs only keep track of the black and gray pages, and a straightforward induction shows that, at any step, there are at most $2k$ gray and black pages.

It is not obvious that the gray algorithm could be implemented to maintain the prescribed set of gray pages without paying a large overhead. The following lemma shows that $\mathcal{G}$ is lazy.

Lemma 5.2 *The gray algorithm $\mathcal{G}$ is lazy.*

Proof. Let σ be a hard sequence for $\mathcal{G}$. Suppose that $p \in \mathcal{M}(\mathcal{G}, j+1) - \mathcal{M}(\mathcal{G}, j)$ and $p \neq \sigma_{j+1}$. First, we show that p is gray at both step j and $j+1$. Since $p \notin \mathcal{M}(\mathcal{G}, j)$, p is not black at step j, and since $p \neq \sigma_{j+1}$, p is not black at step $j+1$. Since $p \in \mathcal{M}(G, j+1)$, p is not white at step complete a rotation. The basic plan of the adversary is as follows: repeatedly request a page that is not in the online algorithm G's fas memory and is farthest from the page where G is positioned. We will assume without loss of generality that G is lazy. Let H be the algorithm that the adversary uses to satisfy the request sequence.

The request sequence: The adversary proceeds in phases. We will maintain that, at the beginning of a phase, G and H hold the same set of pages W in fast memory and that H is positioned on a page q such that $q+1, q+2 \notin W$. In this proof, the set W will take the function that the working set had in the previous proofs, namely, we will extract requests from W in order to make G fault. The adversary starts the phase by requesting page $q+1$ followed by page $q+2$. We will maintain that there are always at least two pages in W that G does not have. The adversary will request one of those pages, and if r is the page where G is currently positioned, the adversary will issue a request for the missing page α that maximizes the quantity $\alpha - r$. However, the adversary will also insert other page requests to force G's fast memory configuration. On the whole, the phase has the form $\langle q+1, q+2\rangle \rho_1 \langle \gamma_1 \rangle \rho_2 \langle \gamma_2 \rangle \ldots \rho_{k-2} \langle \gamma_{k-2} \rangle$ where

- γ_j is the page in W that is not in G's fast memory, and that is farthest from the current position r of G, $1 \le j \le k-2$,
- ρ_j denotes the sequence of requests $\rho_j = (\langle q+1, q+2, \gamma_1, \ldots, \gamma_{j-1}\rangle)^{c+1}$, where $c = \Omega(n \log k)$ and $1 \le j \le k-2$.

Suppose that G and H do not have the same set of pages in fast memory after the request for γ_{k-2}. In that case, the adversary keeps requesting a page that H has, but G does not. Eventually, G and H will have the same set of pages in fast memory, or else G is not competitive, and another phase starts.

Costs of G and H in each phase: Let $R_j = \{q+1, q+2, \gamma_1, \ldots, \gamma_{j-1}\}$ denote the set of pages requested in the sequence ρ_j. Notice that R_j is precisely the set of pages requested since the beginning of the phase until the request γ_j, and that $R_j \subset R_{j+1}$. The algorithm H keeps R_{k-1} throughout the phase and pays a cost of two. We can assume the following about the behaviour of the algorithm G by Lemma 4.5: (i) G has R_j in its fast memory before the request for γ_j, and (ii) that G does not move at all during the ρ_j's, but that it moves only to service the requests for the γ_j's. Thus from here on we can simply assume that a phase merely consists of the sequence of requests $\langle q+1, q+2, \gamma_1, \gamma_2, \ldots, \gamma_{k-2}\rangle$. By this assumption, γ_{j+1} is the page in W that is not in G's fast memory and that maximizes $\gamma_{j+1} - \gamma_j$.

We will now show that the cost of G is $\Omega(n \log k)$ on the sequence $\Gamma = \langle q+1, q+2, \gamma_1, \gamma_2, \ldots, \gamma_{k-2}\rangle$. To estimate G's cost, we will divide a phase into subphases with the property that G executes one rotation per subphase. Then, it will be enough to count the number of subphases. Specifically, we define subsequences $Z_1, Z_2, \ldots, Z_t$ such that

- $Z_1 Z_2 \ldots Z_t = \Gamma$, and
- $Z_1 Z_2 \ldots Z_i$ is the smallest prefix of Γ on which G completes i rotations $(1 \le i < t)$.

Since G executes at least $t-1$ rotations in the phase, it suffices to show that $t = \Omega(\log k)$. The idea is to show that $|Z_i|$ decreases as i increases, and then it will follow that t is large. Let x_i denote the number of pages in W that have not been requested prior to the start of the subsequence Z_i. Then we claim that $|Z_i| \le \lceil (x_i+1)/2 \rceil$. This easily follows from the observations that at each

- $C_j = \{p \in W_i : \beta_j < p\}$.

Notice that $r_j \in A_j \cup B_j \cup C_j$.

We now turn to estimate the transit cost t_j. Define the potential of G immediately before γ_j as

$$\Phi(j) = \begin{cases} \frac{1}{2} & \text{if } r_j \in B_j \cup C_j \\ 0 & \text{otherwise} \end{cases}$$

and the amortized transit cost of G as $t_j + \Phi(j+1) - \Phi(j)$. We will show that the expected amortized transit cost of G during Γ_i is $\Omega(\log k)$. Since the real transit cost of G is equal to the amortized transit cost minus $O(1)$, we obtain that the real transit cost of G during Γ_i is $t = \Omega(\log k)$. Putting together, it will then follow that the actual cost of G during Γ_i is $\Omega(n \log k)$.

If $r_j \in A_j$ and $\beta_j = \gamma_j$, then the potential increases by $1/2$. Moreover, $\beta_j = \gamma_j$ with probability $1/|Y_{ij}| = 1/(k+3-j)$, so that G's expected amortized transit cost when $r_j \in A_j$ is at least $(1/2)/(k+3-j)$. If $r_j \in B_j \cup C_j$ and $\alpha_j = \gamma_j$, then G pays a real transit cost at least equal to 1, and the potential drops by no more than $1/2$. Moreover, $\alpha_j = \gamma_j$ with probability $1/|Y_{ij}| = 1/(k+3-j)$, so that G's expected amortized transit cost when $r_j \in B_j \cup C_j$ is at least $(1/2)/(k+3-j)$. The expected amortized transit cost of G during Γ_i is at least

$$\sum_{j=1}^{k} \left(\frac{1}{2} \frac{1}{k+3-j} \right) = \frac{1}{2} \sum_{j=3}^{k+2} \frac{1}{j} = \Omega(\log k) \ .$$

Consequently, $t = \Omega(\log k)$ and the actual cost of G during Γ_i is $\Omega(n \log k)$.

Finally, we recall that on $\Gamma_1 \Gamma_2 \ldots \Gamma_{(n-k)/2} \Gamma_0$, the cost of H is n, and the cost of G is $\Omega(n^2 \log k)$, which proves the theorem. □

Conversely, the competitive randomized algorithms in [3, 8, 16] immediately imply an $O(n \log k)$-competitive randomized algorithm for BDP even against an ubiquitous adversary.

5 BDP Algorithms with Prefetching

In this section, we will establish our main theorems. We will first show a lower bound of $\Omega(n \log k)$ on the competitive ratio of any deterministic online algorithm that uses prefetching, even when compared to an adversary that does not do any prefetching. We will then present a deterministic online algorithm that achieves a competitive ratio of $O(n \log k)$ and therefore is optimal up to a constant factor.

5.1 A Lower Bound

We first prove the lower bound.

Theorem 5.1 *No deterministic algorithm for broadcast disk paging can achieve a competitive ratio better than $\Omega(n \log k)$ when $3 \leq k \leq bn$ for any fixed $b < 1$ even if the adversary is not allowed to do prefetching.*

Proof. The main difference between the present proof and that of Theorem 4.3 is that G might reload α when it faults on β, and so the online algorithm cannot be made to execute one rotation every other request. In fact, we will typically need several requests to have the online algorithm

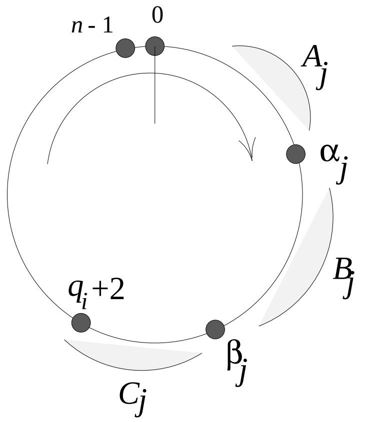

Figure 2: Lower bound for randomized algorithms without prefetching

that there is a page in S_{i-1} missing from G's cache immediately before the request to γ_1. By the previous lemma, G pays at least $c = \Omega(n \log k)$ on ρ_0. Analogously, suppose that there is one of $\gamma_1, \ldots, \gamma_{j-1}$ missing from G's cache immediately before the request to γ_j $(1 \leq j \leq k)$. By the previous lemma, G pays at least $c = \Omega(n \log k)$ on ρ_{j-1}. Therefore, $\Omega(n \log k)$ bounds G's expected cost conditioned to the event that either (i) or (ii) occurred. It remains to show that G's expected cost is $\Omega(n \log k)$ even when neither (i) nor (ii) occurred. We will condition the rest of the analysis to such event, and so we will assume that G has $\gamma_1, \ldots, \gamma_{j-1}$ in fast memory before the request for γ_j where $j = 2, \ldots, k$, and G has all the elements in S_{i-1} before γ_1.

Costs of G and H: The adversary's algorithm H keeps S_i in fast memory throughout Γ_i. Therefore, H pays a cost of at most n on the sequence $\Gamma_1 \ldots \Gamma_{(n-k)/2}\Gamma_0$, exactly as in the previous proofs. We will now show that the expected cost of G on Γ_i $(1 \leq i \leq (n-k)/2)$ is $\Omega(n \log k)$. Let

$$t_j = \begin{cases} 1 & \text{if } G \text{ transits over page 0 while serving } \gamma_j \\ 0 & \text{otherwise} \end{cases}$$

The quantity t_j will be refered to as the *transit cost* of G for γ_j. Define $t = \sum_{j=1}^{k} t_j$ to be the transit cost of G in the phase Γ_i. Observe that G executes at least $t-1$ rotations during Γ_i, and so G's cost during Γ_i is at least $n(t-1)$. Hence, it is enough to show that $t = \Omega(\log k)$.

Before estimating t, we will make some definitions and observations. Let Y_{ij} be the set of pages in W_i that have not been requested prior to the first request for γ_j, that is, $Y_{ij} = W_i - \{\gamma_1, \ldots, \gamma_{j-1}\}$, $j = 1, \ldots, k$. Then by assumptions (i) and (ii) above, there are at least two pages in Y_{ij} that are missing from G's fast memory before γ_j is requested. Let $\alpha_j, \beta_j \in Y_{ij}$ be two such pages and assume without loss of generality that $\alpha_j < \beta_j$, as depicted in Figure 2. Notice that if G does not fault on γ_j, then G does not move and $\alpha_{j+1} = \alpha_j$ and $\beta_{j+1} = \beta_j$. Let r_j be the the position taken by G before the request for γ_j. It can be seen that $r_j \in W_i$ by induction on j. Moreover, G is lazy and so $r_j \neq \alpha_j, \beta_j$. As depicted in Figure 2, let

- $A_j = \{p \in W_i : p < \alpha_j\}$,
- $B_j = \{p \in W_i : \alpha_j < p < \beta_j\}$, and

Proof. By induction on c. The claim is clearly true for $c = 1$. Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_l\}$ and $\sigma' = \langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$. Suppose that $\Sigma - \mathcal{M}(G, lh)) \neq \emptyset$. Since G is lazy, $\mathcal{M}(G)$ will remain unchanged as soon as $\Sigma \subseteq \mathcal{M}(G)$. Then, $\Sigma - \mathcal{M}(G, l(h-1)) \neq \emptyset$, and by induction hypothesis $c(G, (\sigma')^{h-1}) \geq h - 2$. Moreover, G faults at least once during the last repetition of σ', and the lemma follows. □

Theorem 4.6 *No randomized online algorithm without prefetching can have a competitive ratio better than* $\Omega(n \log k)$ *for* $3 \leq k \leq bn$ *and any fixed* $b < 1$*, even if the adversary is not allowed to do prefetching.*

By the minimax theorem [5], it will be enough to show that there is a probability distribution $\mathcal{P}$ over sequences of page requests such that for any lazy deterministic algorithm G, $E_{\mathcal{P}}\{c(G, \sigma)\} \geq c \cdot c(H, \sigma)$ where $c = \Omega(n \log k)$ and H is an optimal offline strategy.

Proof. The main difference between the proofs of Theorem 4.4 and Theorem 4.6 is that the adversary cannot foresee what pages α and β are in the working set, but not in G's fast memory. We sidestep the difficulty by showing that with some probability the adversary is still able to force G to execute one rotation.

The request sequence: The page request sequence σ is of the form $\sigma = (\Gamma_1 \ldots \Gamma_{(n-k)/2} \Gamma_0)^l$, where l is a positive integer, $\Gamma_0 = \langle 0, 1, \ldots, k-1 \rangle$, and Γ_i $(1 \leq i \leq (n-k)/2)$ will be defined later. Roughly speaking, the subsequences Γ_i, for $i \geq 1$, are the analogoues of phases in the previous proofs, and Γ_0 corresponds to the final coordination step that brings about the initial configuration $\{0, 1, \ldots, \triangleright(k-1)\}$. We will now describe the phase Γ_i, a few quantities determined by the Γ_i's, and the value of these quantities at the beginning of Γ_i. A phase Γ_i depends on the set S_{i-1}, which is defined recursively: let $S_0 = \{0, 1, \ldots, k-1\}$, and S_i contains all the pages requested during the phase Γ_i. Throughout we will maintain the following properties:

- $|S_i| = k$ for $1 \leq i \leq (n-k)/2$.
- H holds S_{i-1} in fast memory immediately before the start of phase Γ_i.
- The pages $q_i + 1, q_i + 2 \notin S_{i-1}$ where $q_i = k - 1 + 2i$. In other words, we maintain that $q_i + 1$ and $q_i + 2$ are not in H's fast memory immediately before phase Γ_i starts. For example, k and $k + 1$ are not in H's fast memory immediately before Γ_1.

Define the working set in phase Γ_i to be the set $W_i = S_{i-1} \cup \{q_i + 1, q_i + 2\}$. Let $\gamma_1, \gamma_2, \ldots, \gamma_k$ be a random k-permutation of the working set W_i; define $S_i = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$. The basic plan of the adversary is to request the pages in S_i in the given order $\gamma_1, \gamma_2, \ldots, \gamma_k$. However, the adversary will also request a sequence of other pages between the first request for a page γ_j and the first request for page γ_{j+1}, $1 \leq j \leq (k-1)$ in order to force G's memory configuration. Specifically, $\Gamma_i = \rho_0 \langle \gamma_1 \rangle \rho_1 \langle \gamma_2 \rangle \rho_2 \ldots \langle \gamma_k \rangle$ where

- ρ_0 is a repetition for $c + 1$ times of S_{i-1}, and
- $\rho_j = (\langle \gamma_1, \gamma_2, \ldots, \gamma_j \rangle)^{c+1}$ where $1 \leq j \leq k$.

We will now turn to estimate the expected cost of G during Γ_i conditioned to the event that either (i) a page in S_{i-1} is not in G's cache immediately before the first request to γ_1, or (ii) one of the $\gamma_1, \ldots, \gamma_{j-1}$ is not in G's cache immediately before the first request to γ_j $(2 \leq j \leq k)$. Suppose configuration is $\mathcal{M}(H) = (\mathcal{M}(H) - \{r_1, r_2\}) \cup \{q+1, \triangleright(q+2)\}$. Notice that $r_1, r_2 \in \mathcal{M}(H, j)$ implies $\{r_1, r_2\} \cap \{q+1, q+2\} = \emptyset$, a fact that we use later on.

The rest of the phase is divided into segments. A segment is defined exactly as in the proof of Theorem 4.3 and consists of two page requests that force G to execute one rotation. Both segment requests cause a fault for G, but no fault for H. Therefore, the adversary generates segments while $|\mathcal{M}(H) - \mathcal{M}(G)| \geq 2$. Notice that the adversary is adaptive, and so it can determine if $|\mathcal{M}(H) - \mathcal{M}(G)| \geq 2$, which pages to request in each segment, and in which order. However, the adversary cannot determine at the beginning of a phase the set F of pages that will make G fault. It remains to estimate the expected number of segments. At all steps, G keeps all the pages in the working set W except two. Let α and β be the two pages that are in the working set, but not in G's fast memory immediately after the ith segment has been serviced. Notice that the adversary generates an $(i+1)$st segment only if $|\mathcal{M}(H) - \mathcal{M}(G)| \geq 2$, or, equivalently, only if $\{\alpha, \beta\} \cap \{r_1, r_2\} = \emptyset$. Therefore, G should choose α and β so that $\{\alpha, \beta\} \cap \{r_1, r_2\} \neq \emptyset$. In other words, G should choose the missing pages α and β in order to guess H's missing pages r_1 and r_2. Since $\{r_1, r_2\} \cap \{q+1, q+2\} = \emptyset$, we assume without loss of generality that G always keeps $q+1$ and $q+2$ in fast memory, and so $\{\alpha, \beta\} \cap \{q+1, q+2\} = \emptyset$. We will now estimate the expected number of segments by means of the following simple random experiment. We will have a bin that contains k balls — a white ball for each page in $\mathcal{M}(H, j) - \{r_1, r_2\}$ and a black ball each for r_1 and r_2. In each segment, G extracts two balls α and β from the bin. If both α and β are white, then G gets another round, otherwise, either α or β are black and the experiment stops. The number of segments ℓ is exactly the number of rounds needed to extract a black ball. Clearly, ℓ is minimum if no ball is ever replaced in the bin. Elementary combinatorics now yields $E[\ell] = \Theta(k)$. As an aside, we observe that the case when balls are not replaced in the bin corresponds to the case when G keeps in fast memory a different subset of the working set in different segments.

After the last segment, $|\mathcal{M}(H) - \mathcal{M}(G)| \leq 1$. If $|\mathcal{M}(H) - \mathcal{M}(G)| = 0$, then $\mathcal{M}(G)$ contains the same pages as $\mathcal{M}(H)$, and another phase starts. If $|\mathcal{M}(H) - \mathcal{M}(G)| = 1$, then the adversary keep requesting pages in $\mathcal{M}(H) - \mathcal{M}(G)$ until $\mathcal{M}(G)$ contains the same pages as $\mathcal{M}(H)$. Again, the adversary is adaptive and knows if $\mathcal{M}(G)$ coincides with $\mathcal{M}(H)$. There are two cases, depending on whether this steps ends or not. The expected cost of G conditioned to the event that $\mathcal{M}(G)$ never coincides with $\mathcal{M}(H)$ is infinite, and so G is not competitive. The expected cost of G conditioned to the event that $\mathcal{M}(G)$ eventually coincides with $\mathcal{M}(H)$ is non-negative. We will assume from now on that indeed G's memory will eventually coincide with H's. After $\mathcal{M}(G)$ and $\mathcal{M}(H)$ contain the same pages, a new phase starts. The expected cost of G in a phase is at least equal to the expected cost in the segments, and so it is at least $nE[\ell] = \Omega(nk)$. Meanwhile, the cost of H in the entire phase is due only to the initial faults on $q+1$ and $q+2$, and so it is 2. The adversary issues $(n-k)/2$ request phases paying only a cost of two per phase. Then the adversary pays a cost of k and returns to the starting configuration $\{0, 1, \ldots, \triangleright(k-1)\}$. It will then keep requesting pages $0, 1, \ldots, k-1$ until the fast memory configurations of G and H coincide. Therefore, the cost of the adversary on $(n-k)/2$ phases is n, while the expected cost of G is $\Omega(nk(n-k))$, and the result follows. $\square$

We next establish an $\Omega(n \log k)$ lower bound on the competitive ratio of randomized online algorithms without prefetching against an oblivious adversary. We start with the following simple proposition.

Lemma 4.5 *Let G be a deterministic lazy algorithm, $\sigma = (\langle \sigma_1, \sigma_2, \ldots \sigma_l \rangle)^h$ for some $l \leq k$. If $\{\sigma_1, \sigma_2, \ldots, \sigma_l\} - \mathcal{M}(G, lh) \neq \emptyset$, then $c(G, \sigma) \geq h - 1$ for all configurations $\mathcal{M}(G, 1)$.*

The adversary starts the phase by requesting page $q+1$ followed by page $q+2$. Thus, both G and H are now positioned at $q+2$. The rest of the phase is divided into segments. We will let r be the page where G is positioned at the beginning of a current segment. For example, in the first segment, $r = q+2$. Let α and β be the two pages that are in the working set W, but not in G's fast memory at the beginning of a segment. Assume without loss of generality that $\beta - r > \alpha - r$. Then, the adversary requests β followed by α, and the segment is over. The algorithm G faults on the request for β, transits over α while waiting for β to be transmitted; but it does not load α because it is not allowed to prefetch. Therefore, G faults also on the request for α. On the whole, G transits over α at least twice during the segment, and so it executes at least one rotation per segment. We will now describe how many segments are generated. The adversary counts the number of distinct pages that made G fault during all the segments generated so far. The adversary generates segments until G has faulted on a set F of at least $k-2$ distinct pages. Notice that F is contained in the working set. Moreover, during each segment, G faults on at most two new pages, so that $|F| \in \{k-2, k-1\}$. Hence, at least $\lceil (k-2)/2 \rceil$ segments are generated. We will now specify how H serves the requests in the phase. The algorithm H faults on $q+1$ and $q+2$. At this point, its fast memory contains all the pages of $F \cup \{q+2\}$, plus another arbitrary page from W if $|F \cup \{q+2\}| < k$. Notice that H does not prefetch any page and that H does not fault during any segment. If G and H's fast memory configurations still differ after all segments have been generated, the adversary keeps requesting pages in H's memory that are missing from G's memory, until the two memories coincide in their contents. Notice that if the two memories never coincide, then this step will continue indefinetely, and G is not competitive. Henceforth, assume that G's and H's memory contents will eventually be the same. At this point, the phase is over. The cost of G in the phase is at least the cost in the segment, and so hence at least $n(k-2)/2$. The cost of H in the entire phase is 2 — due only to the initial faults on $q+1$ and $q+2$.

The adversary issues $(n-k)/2$ request phases paying only a cost of 2 per phase. Then the adversary pays a cost of k and returns to the starting configuration $\{0, 1, \ldots, \triangleright(k-1)\}$. It then keeps requesting pages $0, 1, \ldots, k-1$ until the fast memory configurations of G and H coincide. Therefore, the cost of the adversary on $(n-k)/2$ phases is n, while the cost of G is at least $((n-k)/2)(n(k-2)/2)$, and the result follows. □

The construction above extends to a lower bound for randomized algorithms against an adaptive online adversary.

Theorem 4.4 *No randomized online algorithm without prefetching can have a competitive ratio better than $\Omega(nk)$ against an adaptive online adversary for $3 \leq k \leq bn$ and any fixed $b < 1$, even if the adversary is not allowed to do prefetching.*

Proof. The initial setup of the proof is the same as that of Theorem 4.3. The main difference between the proofs of Theorem 4.3 and Theorem 4.4 is that, at the beginning of a phase, the adversary does not know the set F of pages that make G fault. We overcome the difficulty by letting the adversary guess F and showing that the expected number of segments is $\Omega(k)$.

The adversary proceeds in phases. Assume that j is the step of the first phase request, so that $\mathcal{M}(H,j)$ and $\mathcal{M}(G,j)$ denote the configurations of H and G immediately before the current phase begins. We will maintain that $\mathcal{M}(G,j)$ contains the same pages as $\mathcal{M}(H,j)$, and that H is positioned on a page q such that $q+1, q+2 \notin \mathcal{M}(H)$. Define the set $W = \mathcal{M}(H,j) \cup \{q+1, q+2\}$ and call W the working set of the current phase. The adversary starts the phase by requesting $q+1$ followed by $q+2$. Thus, both G and H are now positioned on $q+2$. The algorithm H makes room for $q+1$ and $q+2$ by evicting two random pages r_1 and r_2 from $\mathcal{M}(H,j)$, so that now H's

4 BDP Algorithms Without Prefetching

In this section, we will examine the competitive ratio of online algorithms that do not allow prefetching. In terms of Definition 2.3, such algorithms are lazy for virtual memory paging.

If G is a lazy algorithm for virtual memory paging, it induces an algorithm for BDP that is lazy and whose memory contents after σ_j are exactly the same as G's. For simplicity, we will denote both the virtual memory paging algorithm and its BDP counterpart with the same symbol. A strongly competitive deterministic paging algorithm never incurs more than k times the optimum number of page faults. Since its BDP counterpart is lazy, it follows that a page fault costs at most $n-1$; the last page that was broadcasted and received must be i n the algorithm's cache. So, any strongly competitive paging algorithm, e.g. the marking algorithm [13], is trivially $(k(n-1))$-competitive for broadcast disk paging even against an ubiquitous adversary. In fact, we can show a slightly better competitive ratio.

Theorem 4.1 *Let G be a lazy α-competitive deterministic algorithm for paging. Then, the competitive ratio of G for broadcast disk paging is $(\alpha-1)n+1$.*

Proof. We will assume without loss of generality that the sequence σ is hard for G and that the adversary uses a lazy algorithm H. First, we notice that the cost of H is at least equal to the number of page faults. Indeed, every time H brings a new page into fast memory, H has to transit over that page, paying a unit cost.

Let r_j be the page where the adversary is positioned before the jth request, and define the potential function as $\Phi(j) = r_j - \sigma_{j-1}$. Let $a_j = (\sigma_j - \sigma_{j-1}) + \Phi(j+1) - \Phi(j)$ be the amortized cost of G to serve request σ_j. Notice that if $r_{j+1} = r_j$ then $a_j = (\sigma_j - \sigma_{j-1}) + (r_j - \sigma_j) - (r_j - \sigma_{j-1}) \leq n$. If $\sigma_j \notin \mathcal{M}(H, j)$, then $\Phi(j+1) = 0$ and $a_j = \sigma_j - \sigma_{j-1} + \Phi(j+1) - \Phi(j) = (\sigma_j - r_j) + (r_j - \sigma_{j-1}) - (r_j - \sigma_{j-1})$, which is the real cost of the optimum on that request. Let l be the number of times that the optimum algorithm faults. Then $|\sigma| \leq \alpha l + b$ for some constant b. Moreover, $l \leq c(H, \sigma)$. The resulting cost of G is $c(G, \sigma) = (|\sigma| - l)n + c(H, \sigma) \leq ((\alpha-1)l + b)n + c(H, \sigma) \leq (n(\alpha-1)+1)c(H, \sigma) + bn$, which proves the theorem. □

Corollary 4.2 *The marking algorithm is $((k-1)n+1)$-competitive.*

We next establish an essentially matching lower bound.

Theorem 4.3 *No deterministic online algorithm without prefetching can have a*