LRU is Better than FIFO

Marek Chrobak[*] John Noga[†]

January 15, 1998

Abstract

In the paging problem we have to manage a two-level memory system, in which the first level has short access time but can hold only up to k pages, while the second level is very large but slow. We use competitive analysis to study the relative performance of the two best known algorithms for paging, LRU and FIFO. Sleator and Tarjan proved that the competitive ratio of LRU and FIFO is k. In practice, however, LRU is known to perform much better than FIFO. It is believed that the superiority of LRU can be attributed to locality of reference exhibited in request sequences. In order to study this phenomenon, Borodin, Irani, Raghavan and Schieber [2] refined the competitive approach by introducing the concept of access graphs. They conjectured that the competitive ratio of LRU on each access graph is less than or equal to the competitive ratio of FIFO. We prove this conjecture in this paper.

1 Introduction.

In the paging problem we have a two-level memory system, in which the first level is small and has short access time, while the second level is very large but slow. We will refer to the first memory level as the *cache*, and we denote its size by k. A request to a page p can result either in a *hit*, when p is in the cache, or a *fault*, when it isn't. In response to a fault we need to bring p into the cache. If the cache is already full, some other page q currently in the cache needs to be evicted. The goal of a paging algorithm is to choose the evicted page for each fault so as to minimize the cost, defined as number of faults.

The eviction decisions are made *on-line*, without the knowledge of future requests. It is not difficult to see that no such on-line algorithm can achieve optimal cost on all request sequences. The *competitive ratio* is a performance measure designed to quantify the ability of an on-line algorithm to minimize faults, and is defined as follows: Let $\mathcal{A}$ be an on-line algorithm for paging. We say that $\mathcal{A}$ *is C-competitive* if, for any request sequence ϱ,

$$cost_{\mathcal{A}}(\varrho) \quad \leq \quad C \cdot opt(\varrho) + b, \tag{1}$$

where $cost_{\mathcal{A}}(\varrho)$ is the cost of $\mathcal{A}$ on ϱ, $opt(\varrho)$ is the optimal cost of serving ϱ, and b is a constant independent of ϱ. The competitive ratio of $\mathcal{A}$ is the smallest C for which $\mathcal{A}$ is C-competitive. An algorithm $\mathcal{A}$ will be called *strongly competitive* if its competitive ratio is within a constant factor of the best possible competitive ratio.

Paging has been extensively studied in the literature on competitive on-line algorithms. Belady [1] gave an optimal offline algorithm defined by

[*]Department of Computer Science, University of California, Riverside, CA 92521, USA. Email: `marek@cs.ucr.edu`. Research supported by the NSF grant CCR-9503498.

[†]Department of Mathematics, University of California, Riverside, CA 92521, USA. Email: `jnoga@cs.ucr.edu`. Research supported by the NSF grant CCR-9503498.

BEL: When a fault occurs, evict the page whose next request is furthest in the future.

The two best known on-line algorithms for paging are *Least-Recently-Used* (LRU) and *First-In-First-Out* (FIFO), defined as follows:

LRU: When a fault occurs, evict the page that was accessed least recently.

FIFO: When a fault occurs, evict the page that was brought into the cache least recently.

Sleator and Tarjan [9] proved that the competitive ratio of LRU and FIFO is k.

The competitive approach has been criticized by practitioners for being overly pessimistic. For example, the experiments reported in [10] indicate that the ratio of LRU's cost to the optimal cost on some typical program traces is nearly independent of the cache size. Another problem with competitive analysis is that algorithms with different performance in practice may have the same competitive ratio. The best example of this are LRU and FIFO. In practice, LRU performs much better than FIFO.

One reason for this discrepancy may to be the fact that typical request sequences exhibit *locality of reference*: whenever a page p is requested, the next request is most likely to be chosen from a small set of pages "local" to p (see [1, 3, 7, 8]). It appears that LRU takes advantage of the locality more effectively than FIFO, but competitive analysis does not reflect this superiority of LRU.

These concerns were addressed by Borodin, Irani, Raghavan and Schieber [2] who model the locality of reference by an *access graph* which is known by the online algorithm. The vertices of an access graph G are pages, and G has an edge from p to q if q can be requested immediately after p. As in [2] and [5], we will assume that G is undirected (or, equivalently, that the locality relation is symmetric). We define an algorithm $\mathcal{A}$ to be C-competitive on G if the inequality (1) holds for each ϱ that is a walk in G. The competitive ratio of $\mathcal{A}$ on G, denoted by $C_{\mathcal{A},G}$ is the smallest C for which $\mathcal{A}$ is C-competitive on G. Under this model, LRU is not an optimally competitive algorithm for most graphs. Although it is computationally difficult, an optimally competitive algorithm can be found for any particular finite graph. Both [2] and [6] investigate the problem of finding a natural algorithm which has near optimal competitive ratio. Fiat and Karlin [4] give strongly competitive algorithms for both the deterministic and randomized cases. Since the access graph may not be known by the online algorithm or may require a prohibitive amount of memory to store, Fiat and Mendel [5] define an algorithm as *truly online* if its decisions are based solely on the request sequence. A strongly competitive algorithm for the deterministic, truly online case is given in [5]. Note that both LRU and FIFO are truly online paging algorithms.

One of the problems addressed in [2] is the relative performance of LRU and FIFO. They provide some estimates on the competitive ratios of LRU and FIFO, and they prove that for each access graph G, $C_{\text{LRU},G} \leq 2C_{\text{FIFO},G}$. They also conjecture that the constant 2 can be removed. (This conjecture was also stated in [6].) In other words, the conjecture states that LRU is never worse than FIFO.

In this paper we prove that for all G, $C_{\text{LRU},G} \leq C_{\text{FIFO},G}$, thus settling the conjecture from [2, 6]. Our proof is based on new estimates on the competitive ratios for LRU and FIFO, both of which are of independent interest.

2 Analysis of FIFO and LRU

Throughout the paper the cache size k is an arbitrary but fixed positive integer. By G we will denote a given access graph, and we assume that, without loss of generality, G is connected and has at least $k+1$ vertices.

Both FIFO and LRU can be viewed as keeping the k pages which are in the cache in an ordered list $(s_1, s_2, \ldots, s_k)$. For both algorithms, when a fault p occurs, the last page s_k in the list is removed from the cache and p is placed at the head of the list. When a hit s_i occurs, FIFO does not alter its list, but LRU brings s_i to the head of its list. When FIFO's cache is $(s_1, s_2, ..., s_k)$ then, for $i < j$, s_i entered the cache more recently than s_j. Similarly, when LRU's cache is $(s_1, s_2, ..., s_k)$ then, for $i < j$, s_i was requested more recently than s_j.

Let H be a connected subgraph of G. A vertex v of H is called an *articulation point of* H if the graph $H - v$, obtained from H by removing v, is not connected. By $\alpha(H)$ we denote the number of articulation points of H. Define

$$\alpha_G \;=\; \min_H \alpha(H),$$

where the minimum is taken over connected subgraphs H of G with $k+1$ vertices. Note that $0 \leq \alpha_G \leq k-1$. Also, let

$$\beta_G \;=\; \frac{(k+1)(k-\alpha_G)}{k+1-\alpha_G} \;=\; k - \frac{\alpha_G}{k+1-\alpha_G}.$$

Lemma 1 *For any access graph G, $C_{\text{FIFO},G} \geq \beta_G$.*

Proof: Let H be a connected subgraph of G with vertices $\{s_0, s_1, ..., s_k\}$ which minimizes $\alpha(H)$. Let $\{s_{i_0}, s_{i_1}, ..., s_{i_{l-1}}\}$ be the $l = k+1-\alpha(H)$ non-articulation points of H, where $i_0 = 0 < i_1 < ... < i_{l-1}$. Assume that FIFO's initial cache is $(s_1, s_2, ..., s_k)$. Define algorithm $\mathcal{A}$ to keep all articulation points of H in its cache at all times, and to serve a fault on s_{i_j} by swapping s_{i_j} with $s_{i_{j+1}}$ (the addition $j+1$ is modulo l).

We now define a sequence of requests. If r is the most recent request, $s \in H$ is not in $\mathcal{A}$'s cache, and $s' \in H$ is not in FIFO's cache, then request the vertices along a path from r to s' which does not include s. Such a path must exist because s is not an articulation point of H. When the two caches match, request the vertices along a path from the most recent request to the vertex in H which is in neither cache.

Recall that FIFO's cache can be thought of as an ordered list. Given the way this list is updated on a request and that all requests will be one of the $k+1$ items in H, after any request in this sequence FIFO's cache will have the form $(s_{j+1}, s_{j+2}, ..., s_{j+k})$ where these subscripts will be computed modulo $k+1$.

We now split the request sequence into stages. A stage begins when both FIFO and $\mathcal{A}$ have $(s_1, s_2, ..., s_k)$ in the cache. We will show that the ratio $\mathit{cost}_{\text{FIFO}}/\mathit{cost}_{\mathcal{A}}$ is at least β_G in every stage. For ease of analysis, we further break these stages into l phases, numbered $0, \ldots, l-1$. Phase j begins when both FIFO and $\mathcal{A}$ have s_{i_j} outside the cache. An easy induction argument shows that at the beginning of phase j FIFO's cache will be $(s_{i_j+1}, s_{i_j+2}, ..., s_{i_j+k})$.

Algorithm $\mathcal{A}$ serves phase j at a cost of 1 by swapping s_{i_j} with $s_{i_{j+1}}$. Therefore, the cost of $\mathcal{A}$ in one stage is l.

In serving phase j FIFO will fault on all pages except $\{s_{i_j+1}, s_{i_j+2} ..., s_{i_{j+1}}\}$. So, for any given page in H, FIFO will fault on this page in $l-1$ of the l phases in a stage, and thus the cost of FIFO in one stage is $(k+1)(l-1) = (k+1)(k-\alpha(H))$.

Since $\mathit{cost}_{\mathcal{A}} \geq \mathit{opt}$, by considering the costs in one stage, we have

$$C_{\text{FIFO},G} \;\geq\; \frac{(k+1)(k-\alpha(H))}{k+1-\alpha(H)} \;=\; \beta_G,$$

completing the proof. □

Lemma 2 *For any access graph G, $C_{\text{LRU},G} \leq k - \alpha_G/2$.*

Proof: Let ϱ be the request sequence that is a walk in G. Without loss of generality, initially the caches of LRU and BEL match, and the first request in ϱ is a fault of BEL. Let $r_0, \ldots, r_m$ be the faults of BEL in ϱ, and divide ϱ into phases, $\varrho = \varrho_0 \varrho_1 \ldots \varrho_m$, where each ϱ_i starts with r_i. For $i = 0, \ldots, m+1$, let B_i and L_i denote, respectively, the contents of BEL's cache and LRU's cache after phases $\varrho_0 \ldots \varrho_{i-1}$. Define H_i to be the induced subgraph of G with vertices $B_i \cup \{r_i\}$. Note that H_i has $k+1$ vertices and contains all requests in ϱ_i.

We now define a potential function. Recall that we represent the current content of LRU's cache by the ordered list of the k most recent requests. If $L_i = (l_1, l_2, \ldots, l_k)$ and $d \geq 0$ is the smallest integer for which $\{l_1, l_2, ..., l_{k-d}\} \subseteq B_i$, then let $\Phi_i = \min\{d, \alpha_G/2\}$.

Fix i and let f_i denote the cost of LRU in phase ϱ_i. We will show that

$$f_i + \Phi_{i+1} - \Phi_i \quad \leq \quad k - \frac{\alpha_G}{2}. \tag{2}$$

We consider three cases.

Case 1: $\Phi_i = \alpha_G/2$. Let h be the number of distinct vertices in ϱ_i. Since BEL faults on only the first request of ϱ_i, after serving r_i all vertices in ϱ_i must be contained in BEL's cache. Therefore $h \leq k$. LRU faults at most once on each vertex in ϱ_i, so we have $f_i \leq h$. After phase ϱ_i, the vertices in ϱ_i are the h most recent requests, and they all belong to B_{i+1}. So, we also have $\Phi_{i+1} \leq k - h$. Therefore, inequality (2) holds.

Case 2: $\Phi_i \geq f_i$. Since $f_i + \Phi_{i+1} - \Phi_i \leq \Phi_{i+1} \leq \alpha_G/2 \leq k - \alpha_G/2$, inequality (2) holds.

Case 3: $\Phi_i < \min\{f_i, \alpha_G/2\}$. Since $\Phi_i < \alpha_G/2$, $\Phi_i = d < f_i$. Let $\varrho_i = \sigma\sigma'$, where σ ends with the dth fault of LRU, and let L' be the contents of LRU's cache after σ. If $l \in L'$ then either l has already been requested in σ or $l = l_j$ for some $j \leq k - d$. Therefore, $L' \subseteq H_i$.

We claim that H_i is connected. Since the request sequence is a path, the vertices in L', which are simply the k most recent requests, form a connected subgraph of H_i. Let $H_i - L' = \{s\}$. LRU must fault on σ' at least once, because $f_i > d$. Since all vertices of L' are in the LRU's cache, the first fault in σ' must be on s. Furthermore, the prefix of ϱ_i ending at s is a path in H_i. Thus H_i is connected, and, since $|H_i| = k+1$, we have $\alpha(H_i) \geq \alpha_G$.

We now estimate the number of faults of LRU in σ'. If H_i has an articulation point v, then any connected component of $H_i - v$ has at most $k-1$ vertices. Since LRU's cache is a connected subset of H_i, v will be in LRU's cache during σ'. Therefore, LRU does not fault on articulation points of H_i in σ'. In addition, since the pages in ϱ_i form a connected subgraph of H_i, at least one non-articulation point in H_i cannot be requested in σ'. We conclude that the number of LRU's faults in σ' is $f_i - d \leq |H_i| - 1 - \alpha(H_i) = k - \alpha(H_i)$.

By combining the bounds from the last two paragraphs we obtain $f_i + \Phi_{i+1} - \Phi_i \leq k - \alpha_G + \Phi_{i+1} \leq k - \alpha_G/2$, completing the proof of inequality (2).

Obviously, the cost of BEL in each phase is 1. Summing inequality (2) over all phases yields

$$cost_{\text{LRU}}(\varrho) + \Phi_{m+1} - \Phi_0 \quad \leq \quad \left(k - \frac{\alpha_G}{2}\right) cost_{\text{BEL}}(\varrho).$$

Since $\Phi_{m+1} \geq 0$ and $\Phi_0 = 0$, we get $cost_{\text{LRU}}(\varrho) \leq (k - \alpha_G/2) cost_{\text{BEL}}(\varrho)$, as desired. □

Theorem 1 *For any access graph G, $C_{\text{LRU},G} \le C_{\text{FIFO},G}$.*

Proof: From Lemma 1, Lemma 2, and the fact that $\alpha_G \le k-1$

$$C_{\text{LRU},G} \;\le\; k - \frac{\alpha_G}{2} \;\le\; k - \frac{\alpha_G}{k-\alpha_G+1} \;\le\; C_{\text{FIFO},G}.$$

□

3 Final Comments.

While in the process of showing $C_{\text{FIFO},G} \ge C_{\text{LRU},G}$ for undirected graphs, we have provided improved bounds on the competitive ratios of both LRU and FIFO. For FIFO, Lemma 1 immediately implies the $C_{\text{FIFO},G} \ge (k+1)/2$ lower bound from [2], and gives a much better estimate on $C_{\text{FIFO},G}$ for small values of α_G.

Define

$$a_G \;=\; \max_H \frac{|H| - \alpha(H) - 1}{|H| - k},$$

where the maximum is taken over all connected subgraphs of G with at least $k+1$ vertices. For LRU, the only other general upper bound result we are aware of is $C_{\text{LRU},G} \le 2(a_G + 1)$ from [2]. It can be shown that the bound in Lemma 2 improves on this as long as $\alpha_G \le 2(k+2)/3$.

Borodin, Irani, Raghavan and Schieber [2] proved that $C_{\text{LRU},G} = a_G$ when G is a tree, or when $k = 2, 4$. We believe this equality holds for each access graph. However, except for a few particular cases the exact competitive ratios for both LRU and FIFO are still unknown. It would be nice to be able to extend this analysis to close the gaps. An exact analysis would allow a better consideration of exactly what circumstances allow LRU to greatly outperform FIFO.

Conjecture 1 *For each access graph G, $C_{\text{LRU},G} = a_G$.*

In some situations it may be more natural to have a non-reflexive locality relation. This idea leads to an analogous problem where we now have a directed access graph D. It would be interesting to know whether Theorem 1 holds in this case as well.

Conjecture 2 *For each directed access graph D, $C_{\text{FIFO},D} \ge C_{\text{LRU},D}$.*

Acknowledgments. We would like to thank the anonymous referees for providing useful suggestions that helped us improve the presentation.

References

[1] L.A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5:78–101, 1966.

[2] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.

[3] P. J. Denning. Working sets past and present. *IEEE Trans. Software Eng.*, 6:64–84, 1980.

[4] A. Fiat and A. Karlin. Randomized and multipointer paging with locality of reference. In *Proc. 27th ACM Symposium on Theory of Computing*, pages 626–634, 1995.

[5] A. Fiat and M. Mendel. Truly online paging with locality of reference. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pages 326–335, 1997.

[6] S. Irani, A. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. In *3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 228–236, 1992.

[7] T. Kilburn, D. Edwards, M. Lanigan, and F. Sumner. One-level storage system. *IRE Trans. Elect. Computers*, 37:223–235, 1962.

[8] G. Shedler and C. Tung. Locality in page reference strings. *SIAM Journal on Computing*, 1:218–241, 1972.

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

[10] N. Young. On-line caching as cache size varies. In *Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–250, 1991.