On the relative dominance of paging algorithms[☆,☆☆]

Reza Dorrigiv[*]

Alejandro López-Ortiz

J. Ian Munro

Cheriton School of Computer Science, University of Waterloo, Waterloo, Ont., N2L 3G1, Canada

Abstract

In this paper we give a finer separation of several known paging algorithms using a new technique called relative interval analysis. This technique compares the fault rate of two paging algorithms across the entire range of inputs of a given size rather than in the worst case alone. Using this technique we characterize the relative performance of LRU and LRU-2, as well as LRU and FWF, among others. We also show that lookahead is beneficial for a paging algorithm, a fact that is well known in practice but it was, until recently, not verified by theory.

Key words: Competitive Analysis, Fault Rate, Online Algorithms, Paging

1. Introduction

Paging is a fundamental problem in the context of the analysis of online algorithms. A paging algorithm mediates between a slower and a faster memory. Assuming a cache of size k, the algorithm decides which k memory

[☆]A preliminary version of this paper appeared at ISAAC2007 [1].

[☆☆]This work was supported by NSERC of Canada and the Canada Research Chairs Program.

[*]Corresponding author

Email addresses: rdorrigiv@uwaterloo.ca (Reza Dorrigiv), alopez-o@uwaterloo.ca (Alejandro López-Ortiz), imunro@uwaterloo.ca (J. Ian Munro)

pages to keep in the cache without the benefit of knowing in advance the *sequence* of upcoming page requests. After receiving the ith page request the online algorithm must decide which page to evict, in the event the request results in a fault and the cache is full. The objective is to design efficient online algorithms in the sense that on a given request sequence the total cost, namely the total number of faults, is kept low. Three well known paging algorithms are *Least-Recently-Used* (LRU), *First-In-First-Out* (FIFO), and *Flush-When-Full* (FWF). On a fault, if the cache is full, LRU evicts the page that is least recently requested, FIFO evicts the page that is first brought to the cache, and FWF empties the cache.

The competitive ratio, first introduced formally by Sleator and Tarjan [2], has served as a practical measure for the study and classification of online algorithms. An algorithm (assuming a cost-minimization problem) is said to be α-competitive if the cost of serving any specific request sequence never exceeds α times the optimal cost (up to some additive constant) of an optimal *offline* algorithm which knows the entire sequence. The competitive ratio has been applied to a variety of online problems and settings: it is relatively simple measure to apply yet powerful enough to quantify, to a large extent, the performance of many an online algorithm. On the other hand, it has been observed by numerous researchers [3, 4, 5, 6, 7, 8] that for paging it produces results that are too pessimistic or otherwise found wanting. For example, experimental studies show that LRU has a performance ratio at most four times the optimal offline algorithm [6, 9], as opposed to the competitive ratio k predicted by competitive analysis. Furthermore, it has been empirically well established that LRU (and/or variants thereof) most often are, in practice, preferable paging strategies to all other known paging algorithms [10]. This is in contrast to competitive analysis in which the competitive ratio of LRU is the same as FWF and worse than some randomized algorithms. An additional drawback of competitive analysis, as can easily be shown [11], is that finite lookahead yields no improvement in the performance of an online algorithm. Once again, this is a rather counterintuitive conclusion: in practice, one expects that lookahead should improve performance, and a "reasonable" theoretical measure should reflect this reality.

Such anomalies have been observed since the early days of competitive analysis, and there is a vast literature studying alternative proposals to the competitive analysis of online algorithms in general, and for the paging problem in particular, e.g., [3, 7, 12, 13, 14, 5, 15, 16] (see [17] for a comprehensive survey). Note that competitive analysis uses the concept of an optimal offline algorithm as a baseline for comparing online algorithms. While this may be convenient, it is rather indirect: one could argue that in comparing two online algorithms, all we need to study is the relative cost of the algorithms on the request sequences. The approach we follow in this paper stems from this basic observation. Furthermore, our definition focuses not on a specific worst case request sequence, but rather on the performance of an algorithm on *all* possible sequences.

We briefly review some of these alternatives and refer the reader to the survey of Dorrigiv and López-Ortiz [17] for a more comprehensive and detailed exposition. *Loose competitiveness*, which was first proposed by Young in [6] and later refined in [18], considers an offline adversary that is oblivious to the cache size k. The adversary must then produce a sequence that is bad for most values of k rather than for just a specific value. It also ignores those sequences on which the online algorithm incurs a cost less than a certain threshold. This results in a weaker adversary and gives rise to paging algorithms of constant performance ratio. The *diffuse adversary* model by Koutsoupias and Papadimitriou [5] as well as Young [19, 15] refines the competitive ratio by restricting the set of legal request sequences to those derived from a class (family) of probability distributions. This restriction follows from the observation that although a good performance measure could in fact use the actual distribution over the request sequences, determining the exact distribution of real-life phenomena is a difficult task (e.g., depending on the intended application different distributions might arise). By restricting the input to a class Δ_ϵ of distributions, they are able to show more realistic ratios for the performance of well known paging algorithms. The *Max/Max ratio*, introduced by Borodin and Ben-David [3] compares online algorithms based on their amortized worst-case behaviour (here the amortization arises by dividing the cost of the algorithm over the length of the request sequence). This measure is based on directed comparison of online algorithms and reflects the influence of lookahead. However, it does not provide better separation results than competitive analysis of paging algorithms. The *relative worst order ratio* [7, 16, 20] combines some of the desirable properties of the Max/Max ratio and the *random order ratio* (this last introduced in [21] in the context of the online bin packing problem). As with the Max/Max ratio, it allows for direct comparison of two online algorithms. Informally, this measure compares the performance of two algorithms on a given request sequence by considering the worst-case ordering (permutation) of the sequence, for each algorithm. It then selects among all possible sequences the one that maximizes this worst-case performance. This measure reflects the influence of lookahead for paging and separates the performance of LRU from FWF [16]. More recently, Panagiotou and Souza proposed a model that explains the efficiency of LRU in practice [8]. In their work, they classified request sequences according to some parameters and proved an upper bound on the competitive ratio of LRU as a function of these parameters. Then they argued that, in practice, typical request sequences have parameters that lead to a constant competitive ratio for LRU.

It is well known that “real-life” sequences for paging usually exhibit a high degree of *locality of reference* [11]. This means that the currently requested page is likely to be requested again in the near future. Several theoretical models and techniques have been proposed in order to capture and exploit locality of reference. Borodin, Raghavan, Irani, and Schieber [22] proposed the *access graph* model in which the universe of possible request sequences is reduced to reflect the fact that actual sequences depend heavily on the structure of the program being executed. The space of request sequences can then be modeled by a graph in which paths between vertices correspond to request sequences. Chrobak and Noga showed that the competitive ratio of LRU is at least as good as FIFO on every access graph [23]. In a generalization of the access graph model, Karlin, Phillips, and Raghavan [24] proposed a model in which the space of request sequences has a distribution induced by a Markov chain process. In other work, Becchetti [14] refined the diffuse adversary model of Koutsoupias and Papadimitriou described earlier by considering only probabilistic distributions in which locality of reference is present. Using this model he proves that the performance of LRU improves as locality increases while the reverse is true for FWF. Torng [12] considered the decomposition of request sequences to phases in the same manner as marking algorithms. He then modeled locality of reference by restricting the input to sequences with long average phase length. Using the *full access cost model*, he computed the performance of several paging algorithms on sequences with high locality of reference. He showed that this model reflects the influence of lookahead and also gives constant performance ratios for LRU on sequences with significant locality of reference. However, all conservative and marking algorithms have the same performance in this model.

Albers, Favrholdt, and Giel [13] introduced a model in which request sequences are classified according to a measure of locality of reference. The measure is based on Denning’s working set concept [25] which is supported by extensive experimental results. The technique used reflects the fact that efficient algorithms must perform competitively in each class of inputs of similar locality of reference, as opposed to the worst case alone. Using this model, they showed that LRU has better performance than FWF and FIFO.

These measures achieve different degrees of partial separation between well known algorithms for paging. Recently, Angelopoulos et al. introduced *bijective Analysis* and *average Analysis* [26] which combined with the locality model of Albers et al. [13], shows that LRU is the sole optimal paging algorithm on sequences with locality of reference. They also applied these models to list update algorithms [27]. This resolved an important disparity between theory and practice of online paging algorithms, namely the superiority in practice of LRU.

A remaining question however, is how to characterize the full spectrum of performance of the various known paging algorithms. As discussed above, the competitive ratio focuses on the worst case which in this setting is known to be the same for all algorithms. In this paper we compare instead the performance of two algorithms across the entire range of inputs; in that comparison we use the fault rate measure instead of the competitive ratio. Aside from artifacts introduced by the comparison to an offline algorithm, practitioners find the fault rate a better indicator of performance. Formally, the fault rate of a paging algorithm $\mathcal{A}$ on a sequence σ of length n is the number of faults that $\mathcal{A}$ incurs on $\mathcal{A}$ divided by n. The fault rate of $\mathcal{A}$ on a set of sequences is the worst (maximum) fault rate of $\mathcal{A}$ on any of those sequences. The idea behind the fault rate is that sequences on which $\mathcal{A}$ incurs very few faults compared to the number of requests are not that important, even if the number of faults happens to be much higher than what can be achieved by an offline (or even online) optimum. We show this using an example. Let $\mathcal{A}$ and $\mathcal{B}$ two online paging algorithms so that $\mathcal{A}$ incurs less faults than $\mathcal{B}$ on most sequences. Suppose that the fault rate of $\mathcal{A}$ is generally much lower than that of $\mathcal{B}$, so clearly $\mathcal{A}$ is preferable to $\mathcal{B}$. However, if there happens to be an "easy" sequence σ of length 1000000 on which $\mathcal{A}$ incurs 100 faults, $\mathcal{B}$ incurs 10 faults and optimal offline algorithm can serve σ by only 2 faults, then the competitive ratio of $\mathcal{A}$ is 50 while that of $\mathcal{B}$ is 5 suggesting the opposite of the logical conclusion. Note that the fault rate of $\mathcal{A}$ and $\mathcal{B}$ on σ is 0.01 and 0.001, respectively, which is miniscule and thus of no relevance to the actual performance of a system using either algorithm.

Our results. In this paper we aim to provide a tool for finer study and separation of the relative performance characteristics of online paging algorithms.

Table 1: Summary of the results for relative intervals of several paging algorithms.

	LRU	FWF	FIFO	LIFO	LRU-2
LRU					
FWF	$[0, \frac{k-1}{k}]$				
FIFO	$\supseteq [-\frac{k-1}{k}, \frac{k-1}{2k-1}]$	$[-\frac{k-1}{k}, 0]$			
LIFO	$[-\frac{k-1}{k}, 1]$		$[-\frac{k-1}{k}, 1]$		
LRU-2	$\supseteq [-\frac{k-1}{2k}, \frac{k-1}{k+1}]$				

We propose the *relative interval* which directly compares two online paging algorithms $\mathcal{A}$ and $\mathcal{B}$, without any reference to the optimal offline algorithm. They are compared across their entire performance spectrum (rather than on the worst case alone) using a normalized measure of performance, similar to the fault rate. Informally the relative interval of two algorithms reflects the range of the difference between the fault rate of those algorithms. For every two online paging algorithm $\mathcal{A}$ and $\mathcal{B}$ we define a relative interval $\mathcal{I}(\mathcal{A}, \mathcal{B}) = [\alpha, \beta]$, where $-1 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$. $\beta > -\alpha$, shows that $\mathcal{B}$ is better than $\mathcal{A}$ according to the relative interval. The more the difference, the better $\mathcal{B}$ is compared to $\mathcal{A}$. Table 1 shows the summary of our results about the relative intervals of well-known paging algorithms. These results show that LRU and FIFO are better than FWF, a result expected from practice and experience, yet not fully reflected by the competitive ration model. We also show that the relative interval has another good feature, namely we prove that it reflects the influence of lookahead.

Comparison to other measures. We can directly compare two online algorithms using the relative interval. Other measures that provide direct comparison between online algorithms are the Max/Max ratio, the relative worst order ratio, bijective analysis, and average analysis. The Max/Max ratio reflects the influence of lookahead, but it does not provide better separation than the competitive ratio, e.g., LRU and FWF are equivalent under this measure. The relative worst order ratio reflects the advantage of lookahead and separates the performance of LRU and FWF. Thus this measure gives results comparable to the relative interval. However, it is not intuitive why we should consider all permutations of a sequence for comparing two online paging algorithms (this might be more straightforward for other problems, e.g., bin packing.). Furthermore, the relative interval provides a more comprehensive measure by considering the whole range of possible differences between the performance of two algorithms. Note that in the Max/Max ratio and the relative worst order ratio we only consider the worst case sequence (among all sequences of the same length and all permutations of a sequence, respectively).

Influence of lookahead is reflected by bijective analysis and average analysis. These measures also separate the performance of LRU and FWF. However, all *demand* paging algorithms are equivalent according to bijective analysis and average analysis. A demand paging algorithm does not evict a page on a hit and evicts at most one page on a fault [11]. Thus LRU, FIFO, LRU-2, and LIFO are demand paging algorithms, while FWF is not. Therefore most paging algorithms have the same performance under these measures. However, under the locality of reference model of [13] LRU is the unique optimal deterministic online algorithm under average analysis. This is consistent with the well known fact that LRU is the superior paging algorithm in practice. The relative interval does not reflect the unique optimality of LRU, but this also applies to all other measures that do not incorporate the locality of reference assumption. An interesting extension to this work would be to combine the relative interval with models for locality of reference. We believe that this will lead to better separation results. For example, although LRU beats LIFO under the relative interval, they have close performance. This is not consistent with practice, where LRU behaves much better than LIFO. However, note that LIFO and LRU are equivalent under plain bijective analysis and average analysis. Their performances are only separated when we assume locality of reference. We believe that this would be the case for the relative interval as well.

2. Relative Interval

We introduce a new model for comparing online algorithms. In this model we directly compare two online algorithms, i.e., we do not use the optimal offline algorithm as the baseline of our comparisons. Let $\mathcal{A}$ and $\mathcal{B}$ be two online algorithms for the same minimization problem, e.g., paging. Denote the cost of $\mathcal{A}$ on a sequence σ by $\mathcal{A}(\sigma)$. We define the following two functions:

$$Min_{\mathcal{A},\mathcal{B}}(n) = \min_{|\sigma|=n}\{\mathcal{A}(\sigma) - \mathcal{B}(\sigma)\},$$

and

$$Max_{\mathcal{A},\mathcal{B}}(n) = \max_{|\sigma|=n}\{\mathcal{A}(\sigma) - \mathcal{B}(\sigma)\}.$$

Using these functions we define

$$Min(\mathcal{A},\mathcal{B}) = \liminf_n \frac{Min_{\mathcal{A},\mathcal{B}}(n)}{n}, \qquad \text{and} \qquad Max(\mathcal{A},\mathcal{B}) = \limsup_n \frac{Max_{\mathcal{A},\mathcal{B}}(n)}{n}.$$

Note that $Min(\mathcal{A},\mathcal{B}) = -Max(\mathcal{B},\mathcal{A})$ and $Max(\mathcal{A},\mathcal{B}) = -Min(\mathcal{B},\mathcal{A})$. Now we are ready to define the *relative interval* of $\mathcal{A}$ and $\mathcal{B}$ as

$$\mathcal{I}(\mathcal{A},\mathcal{B}) = [Min(\mathcal{A},\mathcal{B}), Max(\mathcal{A},\mathcal{B})].$$

This interval gives useful information about the relative performance of $\mathcal{A}$ and $\mathcal{B}$. If $Max(\mathcal{A},\mathcal{B}) > |Min(\mathcal{A},\mathcal{B})|$ then $\mathcal{B}$ has better performance than $\mathcal{A}$ in this model. In particular, if $\mathcal{I}(\mathcal{A},\mathcal{B}) = [0,\beta]$ for $\beta > 0$ we say that $\mathcal{B}$ *dominates* $\mathcal{A}$. Note that in this case $\mathcal{A}$ does not have better performance than $\mathcal{B}$ on any sequence (asymptotivcally), while $\mathcal{B}$ outperforms $\mathcal{A}$ on some sequences. Also if $Max(\mathcal{A},\mathcal{B})$ is close to 0 then we can conclude that $\mathcal{A}$ is not much worse than $\mathcal{B}$ on any sequences. We can interpret other situations in an analogous way.

We compute the value of $Min(\mathcal{A},\mathcal{B})$ and $Max(\mathcal{A},\mathcal{B})$ for various choices of $\mathcal{A}$ and $\mathcal{B}$. In some cases we obtained bounds or approximation of these values instead. We say that $[\alpha,\beta]$ approximates the relative interval of $\mathcal{A}$ and $\mathcal{B}$ if $Min(\mathcal{A},\mathcal{B}) \leq \alpha$ and $\beta \leq Max(\mathcal{A},\mathcal{B})$. We show this by $\mathcal{I}(\mathcal{A},\mathcal{B}) \supseteq [\alpha,\beta]$.

3. Relative Interval Applied to Paging Algorithms

In this section we compare some well known paging algorithms using the new model. First we define some other paging algorithms. On a fault, *Last-In-First-Out* (LIFO) evicts the page that is most recently brought to the cache. LIFO does not have a constant competitive ratio [11]. A paging algorithm is called *conservative* if it incurs at most k faults on any sequence that contains at most k distinct pages. It can be shown [11] that LRU and FIFO are conservative algorithms and FWF is not. Another class of online paging algorithms are *marking* algorithms. A marking algorithm $\mathcal{A}$ works in phases. Each phase starts right after the last request of the previous phase and consists of the maximal sequence of requests that contains at most k distinct pages. All the pages in the cache are unmarked at the beginning of each phase. We mark any page just after the first request to it. When an eviction is necessary, $\mathcal{A}$ should evict an unmarked page. It is easy to show that LRU and FWF are marking algorithms while FIFO is not. It is known that the competitive ratio of any conservative or marking paging algorithm is k and this is the best possible among deterministic online algorithms [11]. Therefore LRU, FIFO, and FWF all have the competitive ratio k. LRU-2 is another paging algorithm proposed by O'Neil et al. for database disk buffering [28]. On a fault, LRU-2 evicts the page whose second to the last request is least recent. If there are pages in the cache that have been requested only once so far, LRU-2 evicts the least recently used among them. Boyar et al. proved that LRU-2 has competitive ratio $2k$ [29].

Theorem 1. *For any two online paging algorithms $\mathcal{A}$ and $\mathcal{B}$,*

$$0 \leq Max(\mathcal{A}, \mathcal{B}) \leq 1.$$

Proof. For any n, there is a sequence σ of length n so that $\mathcal{A}(n) = n$, i.e., $\mathcal{A}$ incurs a fault on every request of σ. This sequence can be obtained by requesting, at each step, the page that is evicted by $\mathcal{A}$ in the previous step. $\mathcal{B}$ incurs at most n faults on every sequence of length n. Therefore $\mathcal{B}(\sigma) \leq n$ and $\mathcal{A}(\sigma) - \mathcal{B}(\sigma) \geq 0$. Thus $\max_{|\sigma|=n}\{\mathcal{A}(\sigma) - \mathcal{B}(\sigma)\} \geq 0$. Since this holds for every n, we have $Max(\mathcal{A}, \mathcal{B}) \geq 0$. For the upper bound, note that for every sequence σ of length n, we have

$$\mathcal{A}(n) \leq n \;\Rightarrow\; \mathcal{A}(n) - \mathcal{B}(n) \leq n \;\Rightarrow\; \frac{A(n) - B(n)}{n} \leq 1.$$

Therefore $Max(\mathcal{A}, \mathcal{B}) \leq 1$.

Corollary 1. *For any two online paging algorithms $\mathcal{A}$ and $\mathcal{B}$,*

$$-1 \leq Min(\mathcal{A}, \mathcal{B}) \leq 0.$$

Theorem 2. *Let $\mathcal{A}$ be an arbitrary demand conservative or demand marking algorithm. Then we have $\mathcal{I}(FWF, \mathcal{A}) = [0, \frac{k-1}{k}]$.*

Proof. Let σ be an arbitrary sequence and φ be an arbitrary marking phase of σ. Any marking algorithm incurs at most k faults on φ. Also since φ contains k distinct pages, any conservative algorithm incurs at most k

faults in this phase as well. Thus $\mathcal{A}$ incurs at most k faults on φ. Since FWF incurs exactly k faults on φ we have $Min(FWF, \mathcal{A}) \geq 0$. Then we get $Min(FWF, \mathcal{A}) = 0$ by applying Corollary 1. Now we prove $Max(FWF, \mathcal{A}) \geq \frac{k-1}{k}$ by constructing the following sequence σ which contains $k+1$ distinct pages and starts by $p_1 p_2 \cdots p_k p_{k+1}$. After this the sequence contains several blocks of size k. At the beginning of each block B the cache of FWF contains only one page, say p. Also since $\mathcal{A}$ is a demand paging algorithm its cache contains all of these pages save one (say q). B consists of requests to each of these distinct pages with the exception of p, thus it has length exactly k. In addition, we make sure that the request to q is the last request in B. Thus the last request of B is a fault for both algorithms on which $\mathcal{A}$ evicts one page while FWF flushes its cache and brings only that last request to its cache. Now we can construct a new block in a similar way. Therefore on each block of length k, FWF and $\mathcal{A}$ incur cost k and 1, respectively and we have $Max(FWF, \mathcal{A}) \geq \frac{k-1}{k}$. At each marking phase φ, FWF incurs k faults and $\mathcal{A}$ incurs at least one fault. Also φ has length at least k. Therefore $Max(FWF, \mathcal{A}) \leq \frac{k-1}{k}$.

Theorem 3. *For any conservative algorithm $\mathcal{A}$ and any online algorithm $\mathcal{B}$, we have $Max(\mathcal{A}, \mathcal{B}) \leq \frac{k-1}{k}$.*

Proof. Let σ be an arbitrary sequence of length n and partition σ into blocks so that $\mathcal{B}$ incurs a fault only on the first request of each block. Therefore each block has at most k distinct pages and $\mathcal{A}$ incurs at most k faults in each block. Let $b_1, b_2, \cdots, b_d$ be the sizes of blocks of σ. Then we have $\mathcal{B}(\sigma) = d$ and $\mathcal{A}(\sigma) \leq \sum_{b_i \leq k} b_i + \sum_{b_i > k} k$. Therefore

$$\frac{\mathcal{A}(\sigma) - \mathcal{B}(\sigma)}{n} \leq \frac{\sum_{b_i \leq k} b_i + \sum_{b_i > k} k - d}{\sum_{b_i \leq k} b_i + \sum_{b_i > k} b_i} \leq \frac{\sum_{b_i \leq k}(b_i - 1) + \sum_{b_i > k}(k-1)}{\sum_{b_i \leq k} b_i + \sum_{b_i > k} k}.$$

If $b_i \leq k$, we have $\frac{b_i - 1}{b_i} \leq \frac{k-1}{k}$ and thus $\frac{\sum_{b_i \leq k}(b_i - 1)}{\sum_{b_i \leq k} b_i} \leq \frac{k-1}{k}$. Also we have $\frac{\sum_{b_i > k}(k-1)}{\sum_{b_i > k} k} \leq \frac{k-1}{k}$. Therefore $\frac{\mathcal{A}(\sigma) - \mathcal{B}(\sigma)}{n} \leq \frac{k-1}{k}$. Since this is true for any σ, we have $Max(\mathcal{A}, \mathcal{B}) \leq \frac{k-1}{k}$.

Theorem 4. $\mathcal{I}(LIFO, LRU) = [-\frac{k-1}{k}, 1]$.

Proof. Since LRU is conservative, according to Theorem 3 we have

$$Max(LRU, LIFO) \leq \frac{k-1}{k} \Rightarrow Min(LIFO, LRU) \geq -\frac{k-1}{k}.$$

Now consider the sequence $\sigma = \{p_1 p_2 \cdots p_k p_{k+1}\}^m$. LRU incurs a fault on every request of σ while LIFO incurs a fault on every kth request. Thus

$$Min(LIFO, LRU) \leq -\frac{k-1}{k} \Rightarrow Min(LIFO, LRU) = -\frac{k-1}{k}.$$

For the other direction consider the sequence $\sigma = p_1 p_2 \cdots p_k p_{k+1} \{p_k p_{k+1}\}^m$. LIFO incurs a fault on every request while LRU only incurs a fault on the first k+1 pages. Since m can be arbitrarily large, we have $Max(LIFO, LRU) \geq 1$. According to Theorem 1 we have $Max(LIFO, LRU) = 1$.

A similar argument on the same sequences implies the following theorem.

Theorem 5. $\mathcal{I}(LIFO, FIFO) = [-\frac{k-1}{k}, 1]$.

Theorem 6. $\mathcal{I}(FIFO, LRU) \supseteq [-\frac{k-1}{k}, \frac{k-1}{2k-1}]$.

Proof. $Max(FIFO, LRU)$: Consider the following sequence σ that consists of $k+1$ distinct pages: σ starts with $\sigma_0 = p_1 p_2 \ldots p_k$. After this initial subsequence, σ consists of several blocks. Each block starts right after the previous block and contains $2k-1$ requests to k distinct pages. The first k blocks of σ are shown in Fig. 1. The blocks repeat after this, i.e., the $(k+1)$th block is the same as the first block, the $(k+2)$th block is the same as the second block and so on. It is easy to verify that FIFO incurs a fault on the last k requests of each block while LRU only incurs a fault on the middle request of every block. Let σ have m blocks. Then we have $FIFO(\sigma) = k + m \times k$ and $LRU(\sigma) = k + m$. Therefore

$$\frac{FIFO(\sigma) - LRU(\sigma)}{|\sigma|} = \frac{m(k-1)}{k + m(2k-1)},$$

and for sufficiently large values of m, this value becomes arbitrarily close to $\frac{k-1}{2k-1}$.
$Min(FIFO, LRU)$: Consider the following sequence σ' that consists of $k+1$ distinct pages: σ' starts with $\sigma'_0 = p_1 p_2 \ldots p_k p_{k-1} p_{k-2} \ldots p_1$. After this initial subsequence, σ' consists of m blocks. The first k blocks of σ' are shown in Fig. 2. The blocks repeat after this, e.g., the $(k+1)$th block is the same as the first block. It is easy to verify that LRU incurs a fault on all k requests of

$$\begin{pmatrix}
p_{k-1} & p_{k-2} & \cdots & p_2 & p_1 & \mathbf{p_{k+1}} & p_1 & p_2 & \cdots & p_{k-1} \\
p_{k-2} & p_{k-3} & \cdots & p_1 & p_{k+1} & \mathbf{p_k} & p_{k+1} & p_1 & \cdots & p_{k-2} \\
p_{k-3} & p_{k-4} & \cdots & p_{k+1} & p_k & \mathbf{p_{k-1}} & p_k & p_{k+1} & \cdots & p_{k-3} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
p_k & p_{k-1} & \cdots & p_3 & p_2 & \mathbf{p_1} & p_2 & p_3 & \cdots & p_k
\end{pmatrix}$$

Figure 1: Blocks of sequence σ in the proof of Theorem 6; each row of the matrix represents a block.

$$\begin{pmatrix}
\mathbf{p_{k+1}} & p_k & p_{k-1} & \cdots & p_3 & p_2 \\
\mathbf{p_1} & p_{k+1} & p_k & \cdots & p_4 & p_3 \\
\mathbf{p_2} & p_1 & p_{k+1} & \cdots & p_5 & p_4 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{p_k} & p_{k-1} & p_{k-2} & \cdots & p_2 & p_1
\end{pmatrix}$$

Figure 2: Blocks of sequence σ' in the proof of Theorem 6.

each block while FIFO only incurs a fault on the first request of every block. Then we have $LRU(\sigma) = k + m \times k$ and $FIFO(\sigma) = k + m$. Therefore

$$\frac{FIFO(\sigma) - LRU(\sigma)}{|\sigma|} = \frac{-m(k-1)}{k+mk},$$

and for sufficiently large values of m, this value becomes arbitrarily close to $-\frac{k-1}{k}$.

Theorem 7. *$Max(LRU\text{-}2, LRU) \geq \frac{k-1}{k+1}$.*

Proof. Consider the sequence σ obtained by m times repetition of the block $p_1 p_2 \ldots p_{k-1} p_k p_k p_{k-1} \ldots p_1 p_{k+1} p_{k+1}$. In the first block, LRU incurs $k+1$ faults. In every other block, it only incurs two faults, one on the first request to p_k, and one on the first request to p_{k+1}. Therefore we have $LRU(\sigma) = k + 1 + 2(m-1) = 2m + k - 1$. LRU-2 incurs $k+1$ faults in the first block and $2k$ faults in every other block; it has a hit only on the second requests to p_k and p_{k+1} in each block (other than the first block). Therefore we have $LRU\text{-}2(\sigma) = k + 1 + 2k(m - 1) = 2km - k + 1$. Thus

$$\frac{LRU\text{-}2(\sigma) - LRU(\sigma)}{|\sigma|} = \frac{2km - k + 1 - 2m - k + 1}{m(2k + 2)} = \frac{m(2k - 2) - 2k + 2}{m(2k + 2)},$$

and for sufficiently large values of m, this value becomes arbitrarily close to $\frac{2k-2}{2k+2} = \frac{k-1}{k+1}$.

Theorem 8. *$Max(LRU\text{-}2, LRU) \leq \frac{k-1}{k}$.*

Proof. Let σ be an arbitrary sequence of length n and partition σ to blocks so that LRU incurs a fault only on the first request of each block. Let $B_1, B_2, \ldots, B_d$ be the blocks of σ, and b_i be the size of block B_i. Then we have $LRU(\sigma) = d$ and $LRU\text{-}2(\sigma) \leq \sum_{1 \leq i \leq d} b_i$. We show that LRU-2 incurs at most k faults in each block. We first show B_i contains at most k distinct pages. B_1 contains requests to one page and LRU-2 incurs one fault on it. Consider an arbitrary block B_i for $i > 1$, let p be the first request of B_i, and let p_1, p_2, $\ldots$, p_{k-1} be the $k - 1$ most recently used pages before the block B_i in this order. We have $p \notin P = \{p_1, p_2, \ldots, p_{k-1}\}$, because LRU incurs a fault on p. We claim that each request of B_i is either to p or to a page of P. Assume for the sake of contradiction that B_i contains a request to a page $q \notin \{p\} \cup P$ and consider the first request to q in B_i. All pages $p, p_1, p_2, \ldots, p_{k-1}$ are requested since the previous request to q. Therefore at least k distinct pages are requested since the last request to q and LRU incurs a fault on q. This contradicts the definition of a block. Therefore B_i contains at most k distinct pages.

We claim that LRU-2 incurs at most one fault on every page q in block B_i. Assume that this is not true and LRU-2 incurs two faults on a page q in B_i. Therefore q is evicted after the first request to it in B_i. Assume that this eviction happened on a fault on a request to page r and consider the pages that are in LRU-2's cache just before that request. Since $r \in \{p\} \cup P$ is not in the cache and $|\{p\} \cup P| = k$, there is a page $s \notin \{p\} \cup P$ in the cache. Let t be the time of the last request to p_{k-1} before the block B_i. The last request to s is before t, while the second to last request to q is at time t or afterwards. Therefore LRU-2 does not evict q on this fault, which is a contradiction. Hence LRU-2 incurs at most k faults in each block of σ.

Therefore

$$\begin{aligned}\frac{LRU\text{-}2(\sigma) - LRU(\sigma)}{n} &\le \frac{\sum_{b_i \le k} b_i + \sum_{b_i > k} k - d}{\sum_{b_i \le k} b_i + \sum_{b_i > k} b_i} \\ &\le \frac{\sum_{b_i \le k}(b_i - 1) + \sum_{b_i > k}(k-1)}{\sum_{b_i \le k} b_i + \sum_{b_i > k} k}.\end{aligned}$$

If $b_i \le k$, we have $\frac{b_i - 1}{b_i} \le \frac{k-1}{k}$ and thus

$$\frac{\sum_{b_i \le k}(b_i - 1)}{\sum_{b_i \le k} b_i} \le \frac{k-1}{k}.$$

Also we have

$$\frac{\sum_{b_i > k}(k-1)}{\sum_{b_i > k} k} \le \frac{k-1}{k}.$$

Therefore

$$\frac{LRU\text{-}2(\sigma) - LRU(\sigma)}{n} \le \frac{k-1}{k}.$$

Since this is true for any σ, we have $Max(LRU\text{-}2, LRU) \le \frac{k-1}{k}$.

Theorem 9. *$Min(LRU\text{-}2, LRU) \le -\frac{k-1}{2k}$.*

Proof. Consider the following sequence σ that consists of $k+1$ distinct pages. σ starts with $\sigma_0 = p_1 p_2 \ldots p_k$. After this initial subsequence, σ consists of m blocks. Each block starts right after the previous block. The ith block consists of one of the subsequences shown in Figure 3, depending on the parity of i. It is easy to verify that LRU incurs a fault on the last k requests of each block while LRU-2 only incurs a fault on the middle request of every block, i.e., p_{k+1} in *Odd* blocks and p_1 in *Even* blocks. Then we have $LRU(\sigma) = k + m \times k$ and $LRU\text{-}2(\sigma) = k + m$. Therefore

$$\frac{LRU\text{-}2(\sigma) - LRU(\sigma)}{|\sigma|} = \frac{-m(k-1)}{k + m(2k)},$$

and for sufficiently large values of m, this value becomes arbitrarily close to $-\frac{k-1}{2k}$.

Odd:	p_k	p_{k-1}	$\dots$	p_2	p_1	p_{k+1}	p_k	p_{k-1}	$\dots$	p_3	p_2
Even:	p_2	p_3	$\dots$	p_k	p_{k+1}	p_1	p_2	p_3	$\dots$	p_{k-1}	p_k

Figure 3: A block of sequence σ in the proof of Theorem 9.

Therefore while *Max(LRU-2,LRU)* is almost 1, we have proven so far an upper bound of almost -1/2 for *Min(LRU-2,LRU)*. A natural question is whether we can improve this bound, i.e., prove that *Min(LRU-2,LRU)* is less than -1/2. We believe that this is not true and prove it for the case that we only have $k+1$ distinct pages (note that all our examples are using $k+1$ distinct pages). While this is a counterintuitive result, in the sense that LRU-2 is preferable in practice it adds to our understanding of the relative advantages of LRU and LRU-2. This results indicates that in the fault rate model LRU is also preferable to LRU-2 and hence additional assumptions need to be made in a model (such as the independency of requests model [30]) that would accurately reflect the superiority of LRU-2 observed in practice.

Theorem 10. *If we have at most $k{+}1$ distinct pages then Min(LRU-2,LRU)$\geq -1/2$.*

Proof. We call a request a "disparity" if it is a fault for LRU and a hit for LRU-2. Note that only a disparity may reduce the value of *Min(LRU-2,LRU)*. Consider an arbitrary sequence $\sigma = \sigma_1\sigma_2 \dots \sigma_n$ and an arbitrary page p. Let S be the set of all k distinct pages other than p. We prove that between any two request for p in σ causing a disparity there should be a request to p that is not a disparity. Assume for the sake of contradiction that this is not the case: σ_a and σ_b are disparity requests to p, and there is no request to p between them. Let σ_x be the last request to p before σ_a. Since p_a is a fault for LRU, it has evicted p between p_x and p_a. Therefore all members of S are requested between p_x and p_a. Similarly all pages of S are requested between p_a and p_b. Since p is at LRU-2's cache right before p_a, there should be at least one page in S that is not in its cache at that time. As all pages of S are requested between p_a and p_b, LRU-2 incurs at least one fault in this interval. Let p_y be the last request between p_a and p_b on which LRU-2 incurs a fault. We claim that LRU-2 should evict p on the request p_y. Assume that LRU-2 evicts a page $q \in S$ on the fault p_y. The next request to q would be a fault for LRU-2 and since p_y is its last fault between p_a and p_b and q is requested in this range, we conclude that q has been requested between p_a

and p_y. However this means that the second last request to q is after p_x, while the second last request to p is at p_x. Thus LRU-2 should evict p at p_y, and p_b is a fault for LRU-2 which is a contradiction. Hence corresponding to each request that may reduce the value of *Min(LRU-2,LRU)* there is at least one request that does not. This proves the bound of $-1/2$ for *Min(LRU-2,LRU)*.

Influence of lookahead

We demonstrate that the relative interval reflects the effects of lookahead. Let LRU(ℓ) be a modification of LRU defined for a lookahead of size ℓ as follows [31]. On a fault, LRU(ℓ) evicts a page in the cache that is least recently used among the pages that are not in the current lookahead. It is known [26] that LRU(ℓ) incurs no more faults than LRU on any sequence. Therefore $Min(LRU, LRU(\ell)) = 0$. Now consider the sequence $\sigma = \{p_1 p_2 \dots p_k p_{k+1}\}^m$. LRU incurs a fault on every request of σ while LRU(ℓ) incurs a fault on every lth request. Hence

$$Max(LRU, LRU(\ell)) \geq 1 - 1/l,$$

and thus LRU(ℓ) dominates LRU.

4. Conclusions and Open Questions

We introduced a fault rate based metric to compare paging algorithms and using this metric, we showed the superiority of LRU and FIFO over FWF. The metric also reflects the beneficial influence of lookahead.

Several natural open questions remain: filling in the remaining entries in Table 1 as well as refining the bounds that are not tight. Additionally we believe that the relative interval can be of interest in other online settings and even perhaps for the comparison of offline algorithms.

Acknowledgements. We thank Spyros Angelopoulos for insightful discussions.

References

[1] R. Dorrigiv, A. López-Ortiz, J. I. Munro, On the relative dominance of paging algorithms, in: Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC '07), 2007, pp. 488–499.

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

[3] S. Ben-David, A. Borodin, A new measure for the study of on-line algorithms, Algorithmica 11 (1994) 73–91.

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

[5] E. Koutsoupias, C. Papadimitriou, Beyond competitive analysis, SIAM J. Comput. 30 (2000) 300–317.

[6] N. E. Young, The k-server dual and loose competitiveness for paging, Algorithmica 11 (6) (1994) 525–541.

[7] J. Boyar, L. M. Favrholdt, The relative worst order ratio for online algorithms, ACM Transactions on Algorithms 3 (2).

[8] K. Panagiotou, A. Souza, On adequate performance measures for paging, in: Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06), 2006, pp. 487–496.

[9] R. L. Sites, A. Agarwal, Multiprocessor cache analysis using ATUM, in: Proceedings of the fifteenth Annual International Symposium on Computer Architecture (ISCA '88), 1988, pp. 186–195.

[10] A. Silberschatz, P. B. Galvin, G. Gagne, Operating System Concepts, John Wiley & Sons, 2002.

[11] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, 1998.

[12] E. Torng, A unified analysis of paging and caching, Algorithmica 20 (2) (1998) 175–200.

[13] S. Albers, L. M. Favrholdt, O. Giel, On paging with locality of reference, Journal of Computer and System Sciences 70.

[14] L. Becchetti, Modeling locality: A probabilistic analysis of LRU and FWF, in: 12th Annual European Symposium on Algorithms (ESA '04), 2004, pp. 98–109.

[15] N. E. Young, On-line paging against adversarially biased random inputs, Journal of Algorithms 37 (1) (2000) 218–235.

[16] J. Boyar, L. M. Favrholdt, K. S. Larsen, The relative worst-order ratio applied to paging, Journal of Computer and System Sciences 73 (5) (2007) 818–843.

[17] R. Dorrigiv, A. López-Ortiz, A survey of performance measures for on-line algorithms, SIGACT News (ACM Special Interest Group on Automata and Computability Theory) 36 (3) (2005) 67–81.

[18] N. E. Young, On-line file caching, Algorithmica 33 (3) (2002) 371–383.

[19] N. E. Young, Bounding the diffuse adversary, in: Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA '98), 1998, pp. 420–425.

[20] J. Boyar, P. Medvedev, The relative worst order ratio applied to seat reservation, ACM Transactions on AlgorithmsTo appear.

[21] C. Kenyon, Best-fit bin-packing with random order, in: ACM-SIAM SODA '96, 1996, pp. 359–364.

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

[23] M. Chrobak, J. Noga, LRU is better than FIFO, Algorithmica 23 (2) (1999) 180–185.

[24] A. R. Karlin, S. J. Phillips, P. Raghavan, Markov paging, SIAM Journal on Computing 30 (3) (2000) 906–922.

[25] P. J. Denning, The working set model for program behaviour, Communications of the ACM 11 (5).

[26] S. Angelopoulos, R. Dorrigiv, A. López-Ortiz, On the separation and equivalence of paging strategies, in: Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA '07), 2007, pp. 229–237.

[27] S. Angelopoulos, R. Dorrigiv, A. López-Ortiz, List update with locality of reference, in: Proceedings of the 8th Latin American Symposium on Theoretical Informatics (LATIN '08), 2008, pp. 399–410.

[28] E. J. O'Neil, P. E. O'Neil, G. Weikum, The LRU-K page replacement algorithm for database disk buffering, in: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 1993, pp. 297–306.

[29] J. Boyar, M. R. Ehmsen, K. S. Larsen, Theoretical evidence for the superiority of LRU-2 over LRU for the paging problem, in: Proceedings of the 4th Workshop on Approximation and Online Algorithms (WAOA '06), 2006, pp. 95–107.

[30] N. Megiddo, D. S. Modha, ARC: A self-tuning, low overhead replacement cache, in: Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03), 2003, pp. 115–130.

[31] S. Albers, On the influence of lookahead in competitive paging algorithms, Algorithmica 18 (3) (1997) 283–305.