Relative Interval Analysis of Paging Algorithms on Access Graphs[☆]

Joan Boyar[a], Sushmita Gupta[b], Kim S. Larsen[a,1]

[a]*Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*
[b]*Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan*

Abstract

Access graphs, which have been used previously in connection to competitive analysis and relative worst order analysis to model locality of reference in paging, are considered in connection with relative interval analysis. The algorithms LRU, FIFO, FWF, and FAR are compared using the path, star, and cycle access graphs. In this model, some of the results obtained are not surprising. However, although LRU is found to be strictly better than FIFO on paths, it has worse performance on stars, cycles, and complete graphs, in this model. We solve an open question from [Dorrigiv, López-Ortiz, Munro, 2009], obtaining tight bounds on the relationship between LRU and FIFO with relative interval analysis.

Key words: Online algorithms, Paging, Access graphs, Relative interval analysis

---

[☆]A preliminary version of this paper appeared in the proceedings of the Thirteenth Algorithms and Data Structures Symposium, 2013. Supported in part by the Danish Council for Independent Research and the Villum Foundation. Part of this work was carried out while the first and third authors were visiting the University of Waterloo.

Email addresses: `joan@imada.sdu.dk` (Joan Boyar), `sushmita.gupta@gmail.com` (Sushmita Gupta), `kslarsen@imada.sdu.dk` (Kim S. Larsen)

[1]Corresponding author.

1. Introduction

The paging problem is the problem of maintaining a subset of a potentially very large set of pages from memory in a significantly smaller cache. This problem, originating in the design of operating systems, has received much attention in online algorithms research, giving rise to many algorithms and techniques for analyzing them. This attention is largely due to the importance of the problem combined with the dissatisfaction with the results obtained using standard competitive analysis, which is unable to distinguish between many paging algorithms of varying quality.

The paging problem considered in online algorithms is the following: We have a memory that can hold N pages and a cache that can hold $k < N$. A paging algorithm must process a sequence of requests. Each request is a reference to a page in memory, which may or may not be present in the cache. If it is present, this is referred to as a "hit" and otherwise we have a "fault". The paging algorithm must bring the requested page into cache if it is not already there. Since the cache is smaller than the memory, this means that when a page not currently present in cache must be brought in, some other page must be evicted. The only decision a paging algorithm can make is which page to evict. For this reason, paging algorithms are also sometimes referred to as eviction strategies. The problem is online, meaning that when the paging algorithm processes a given request, it has no information about any future requests, and the objective is to minimize the total number of faults. We make the common assumption that the cache is empty to start with, so no evictions have to be carried out for the first k distinct page requests.

1.1. Paging Algorithms

Many different paging algorithms have been considered in the literature, many of which can be found in [3, 12]. Among the best known are LRU (least-recently-used), which always evicts the least recently used page, and FIFO (first-in-first-out), which evicts pages in the order they entered the cache. We also consider a known bad algorithm, FWF (flush-when-full), which is often used for reference, since quality measures ought to be able to determine at the very least that it is worse than the other algorithms. If FWF encounters a fault with a full cache, it empties its cache, and brings the new page in.

Finally, we consider a more involved algorithm, FAR^G [4], which works with respect to a known access graph, G. The access graph is an undirected graph with pages as vertices, which is used to limit the input sequences that are considered. The request sequence can start with any page, but for any point in the request sequence, the next request, if not identical to the previous, must be to a page connected to the current request by an edge in the access graph, i.e., the pages are adjacent. The intention is to provide a model for locality of reference.

The access graph is fixed before the request sequence starts and FAR^G uses this graph as well as marks that it administers on the pages in cache to make its decisions as follows: Whenever a page is requested, it is marked. On a fault, where it is necessary to evict a page, it always evicts an unmarked page. If all pages are marked in such a situation, FAR^G first unmarks all pages. The unmarked page it chooses to evict is the one farthest from any marked page in the access graph G. For breaking possible ties, when more than one unmarked page in cache has the maximal distance to marked pages, we use the LRU strategy in this paper, i.e., evicting the least recently used among these pages.

Other algorithms based on access graphs were investigated in [15, 16, 17].

1.2. Separation Results via Performance Measures

Comparing the behavior of paging algorithms under various assumptions has been a topic for much research. The most standard measure of quality of an on-line algorithm, competitive analysis [22, 18], cannot directly distinguish between most of them. The competitive ratio is inspired by the approximation ratio and is defined as follows: Letting $\mathrm{ALG}(I)$ denote the cost (number of faults) of running an algorithm ALG on the input sequence I, ALG is c-competitive if there exists a constant b such that $\mathrm{ALG}(I) \leq c\,\mathrm{OPT}(I) + b$ for all I. Here OPT is an optimal offline algorithm. Thus, OPT knows the entire input sequence in advance, and its performance is a lower bound on how well any online algorithm can perform. The competitive ratio is the infimum over all c for which the algorithm is c-competitive. Competitive analysis deems LRU, FIFO, and FWF equivalent, with a competitive ratio of k [21, 18], where k denotes the size of the cache.

Other measures, such as bijective/average analysis [2], relative worst order analysis [5, 6], and relative interval analysis [13] can be used to obtain more separations. For the discussions here and later in the introduction, we briefly define these measures without introducing extra notation. More mathematically-based definitions, variations, and refinements of the measures can be found in the respective papers.

As opposed to competitive analysis, bijective analysis is based on a direct comparison between two algorithms, ALG and ALG$'$, rather than via a comparison to OPT. ALG is at least as good as ALG$'$ if for all n, when considering all possible different input sequences of length n, there exists a bijection σ on this set such that for all input sequences I of length n, $\text{ALG}(I) \leq \text{ALG}'(\sigma(I))$. Average analysis simply compares the average performance of the algorithms, again separately for each input length.

In relative worst order analysis, comparisons are also between two algorithms directly, but based on a partition of all possible input sequences. Using the notation from above, ALG is at least as good as ALG$'$ if for all input sequences I, $\text{ALG}_W(I) \leq \text{ALG}'_W(I)$, where $\text{ALG}_W(I)$ denotes ALG's performance on the permutation of I that gives the worst result (most faults).

Relative interval analysis considers the interval spanned by the fraction $\frac{\text{ALG}(I)-\text{ALG}'(I)}{n}$ over all input sequences I of length n. ALG dominates ALG$'$ if the right-most point of the interval is at most zero.

In contrast to competitive analysis, bijective, average, relative worst, and relative interval analysis all establish that LRU and FIFO are better than FWF, and also that look-ahead helps. Here look-ahead means that the strict online constraint that no information about the future is available is weakened, and an algorithm is allowed to see the l next requests for some constant l.

Since LRU performs better than FIFO in some practical situations [24], there has been considerable effort to explain this. To our knowledge, no techniques have been able to separate LRU and FIFO, without adding some modelling of locality of reference. In [20, 25], there are proofs of optimality for LRU against a diffuse adversary, but as far as we can determine, no proof of non-optimality for FIFO.

One explanation of the difficulty in separating LRU and FIFO without considering locality of reference is as follows: If one considers all sequences of length n for any n, bijective/average analysis shows that their average number of faults on these sequences is identical [2], since both are demand paging algorithms.

1.3. Separation Results via Locality of Reference

A separation between FIFO and LRU was established quite early using access graphs for modelling locality of reference [9], showing that under competitive analysis, no matter which access graph one restricts to, LRU always does at least as well as FIFO. This proved a conjecture in [4], where the access graph model was introduced. Another way to restrict the input sequences was investigated in [1]. Using Denning's working set model [10, 11] as an inspiration, sequences were limited with regards to the number of distinct pages in a sliding window of size k. This also favors LRU, as does bijective analysis [2], using the same locality of reference definition as [1]. There has also been work in the direction of probabilistic models, including the diffuse adversary model [20] and Markov chain based models [19].

The earlier successes and the generality of access graphs, together with the possibilities the model offers with regards to investigating specific access patterns, make it an interesting object for further studies. Given the abundance of performance measure for online algorithms, with no clear acceptance of one being better than the others for all problems, it seems natural to explore access graphs results in the context of other performance measures. This has the potential of expanding our understanding of the relative strengths of various performance measures as well as concrete algorithms, a direction of work initiated in [8].

One step in that direction was carried out in [7], where more nuanced results were demonstrated, showing that restricting input sequences using the access graph model, while applying relative worst order analysis, LRU is strictly better than FIFO on paths and cycles. The question as to whether or not LRU is at least as good as FIFO on all finite graphs was left as an open problem, but it was shown that there exists a family of graphs which grows with the length of the corresponding request sequence, where LRU and FIFO are incomparable. Since LRU is optimal on paths, it is not surprising that both competitive analysis and relative worst order analysis find that LRU is better than FIFO on paths. Any "reasonable" analysis technique should give this result. Under competitive analysis, LRU and FIFO are equivalent on cycles. The separation by relative worst order analysis occurs because cycles contain paths, LRU is better on paths, and relative worst order analysis can capture this. The fact that there exists an infinite family of graphs which grows with the length of the sequence where LRU and FIFO are incomparable is not particularly surprising or disturbing. There are many sequences where FIFO is better than LRU; they just seem to occur less often in real applications.

1.4. Previous Work on Relative Interval Analysis

In some ways relative interval analysis is between competitive analysis and relative worst order analysis. As with relative worst order analysis, two algorithms are compared directly to each other, rather than compared to OPT. This gives the advantage that, when one algorithm dominates another, in the sense that it is at least as good as the other on every request sequence and better on some, the analysis will reflect this. On the other hand, it is similar to competitive analysis in that the two algorithms are always compared on exactly the same sequence. It is different from both competitive analysis and relative worst order analysis, however, in that the result is not a ratio. To compare two algorithms, LRU and FIFO for example, one considers the difference between LRU's and FIFO's performance on any sequence, divided by the length of that sequence. The range that these ratios can take is the "interval" for that pair of algorithms. As opposed to a worst-case measure such as competitive analysis, relative interval analysis can convey more detailed information. For instance, it can reflect that one algorithm ALG can be much better than another algorithm ALG$'$ on some sequences, but no matter which sequences one considers, ALG can only be slightly worse than ALG$'$.

For FIFO and LRU, [13] found two families of sequences I_n and J_n such that $\lim_{n\to\infty} \frac{\mathrm{FIFO}(I_n)-\mathrm{LRU}(I_n)}{n} = -1 + \frac{1}{k}$ and $\lim_{n\to\infty} \frac{\mathrm{FIFO}(J_n)-\mathrm{LRU}(J_n)}{n} = \frac{1}{2} - \frac{1}{4k-2}$. They left it as an open problem to determine if worse sequences exist, making the interval even larger. In their notation, they proved: $[-1 + \frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}] \subseteq \mathcal{I}(\mathrm{FIFO}, \mathrm{LRU})$.

1.5. Our Results and Their Relations to Previous Results

We start by resolving the open problem from [13], showing that their comparison between LRU and FIFO can be made tight: $\mathcal{I}(\mathrm{FIFO}, \mathrm{LRU}) = [-1 + \frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}]$. Following [13], these results would be interpreted as saying that FIFO has *better performance* than LRU, since the absolute value of the minimum value in the interval is larger than the maximum, but also that they have different strengths, since zero is contained in the interior of the interval.

We obtain more nuanced results by considering various types of access graphs, such as paths (P_N), stars (S_N), and cycles (C_N), splitting the interval of $[-1 + \frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}]$ into subintervals for the respective graph classes. When the access graph is complete (a clique), there are no restrictions on the input sequences, so this is equivalent to considering the situation without an access graph.

This paper contains a systematic treatment of all four algorithms (LRU, FIFO, FWF, FAR) on all four classes of graphs (general, paths, stars, cycles), comparing each pair of these algorithms to each other. At the end of the paper, in Table 1, we give a technical overview of our results that may be a useful reference point after and while reading the paper, past the preliminary section with the relevant notation. Here, we discuss the main results that can be extracted from the table.

Comparing our results with the results from competitive analysis and relative worst order analysis, both with respect to access graphs, it becomes clear that different measures highlight different aspects of the algorithms. All the measures show that LRU is strictly better than FIFO on paths. This is not surprising since LRU is in fact optimal on paths, performing as well as the optimal offline algorithm [4], while FIFO is not.

In contrast to the relative worst order results that also favored LRU, on cycles and complete graphs, relative interval analysis gives results which can be interpreted as incomparability, but leaning towards deeming FIFO the better algorithm. More precisely, if one considers Table 1, for large cache sizes k, the interval representing the comparison between FIFO and LRU on star graphs tend to $[-\frac{1}{2}, \frac{1}{4}]$, and to $[-1, \frac{1}{2}]$ on cycles, i.e., in both cases, FIFO sometimes has a more significant advantage over LRU than LRU ever has over FIFO.

This difference between relative interval analysis and relative worst order analysis seems to be due to the fact that relative interval analysis uses the absolute difference in the fault rate (number of faults divided by the length of the sequence). Thus, there is a normalization relative to the lengths of the sequences. In order for LRU to perform differently from FIFO on a sequence, there must be requests to pages which are currently in cache, adding to the length of the sequence. For some access graphs, a longer sequence is necessary for LRU to perform significantly better than FIFO than is necessary for FIFO to perform better than LRU.

Since LRU is generally better than FIFO in practice, this indicates that relative interval analysis lacks some of the predictive power the other two measures have with respect to the paging problem, even when the access graph technique is added. However, relative interval analysis has the advantage over competitive analysis that it correctly predicts that FWF is worse than LRU on general graphs and that look-ahead helps.

Finally, it turns out to be quite cumbersome and notationally heavy to analyze FAR on cycles. Also here, we get results which can be interpreted as incomparability, but leaning very slightly towards deeming FAR the better algorithm when comparing it to LRU and FIFO. It would be interesting to know if the difference could be more pronounced on more complicated graph structures, but we fear that analyses will become too complicated.

2. Preliminaries

We have defined the paging algorithms in the introduction. If more detail is desired, the algorithms are described in [3].

An *access graph* for paging models the access patterns, i.e., which pages can be requested after a given page. Thus, the vertices are pages, and after a page p has been requested, the next request is to p or one of its neighbors in the access graph. We let N denote the number of vertices of the access graph under consideration at a given time and this is the same as the number of different pages we consider. We assume that pages are numbered from 1 through N. We will always assume that $N > k$, where k is the cache size, since otherwise essentially all algorithms are equivalent. A request sequence is a sequence of pages and the sequence *respects* a given access graph if any two consecutive requests are either identical or neighbors in the access graph. We let $\mathcal{L}(G)$ denote the set of all request sequences respecting G.

We use the definition of k-phases from [3]:

Definition 1. A request sequence can be divided recursively into a number of *k-phases* as follows: Phase 0 is the empty sequence. For every $i \geq 1$, Phase i is a maximal sequence following Phase $i-1$ containing at most k distinct requests. $\square$

In other words, Phase i begins on the $(k+1)$st distinct page requested since the start of Phase $i-1$, and the last phase may contain fewer than k different pages. We generally want to ignore Phase 0, and refer to Phase 1 as the first phase.

Similarly, we can define x-blocks, for some integer x, as maximal sequences for which a given algorithm $\mathcal{A}$ has faulted x times:

Definition 2. A request sequence can be divided recursively into a number of *x-blocks* with respect to an algorithm $\mathcal{A}$ as follows: The 0th x-block is the empty sequence. For every $i \geq 1$, the ith x-block is a maximal sequence following the $(i-1)st$ x-block for which $\mathcal{A}$ faults at most x times.

The *complete* blocks are defined to be the ones with x faults, i.e., excluding the 0th block and possibly the last. $\square$

There are some well-known and important classifications of paging algorithms, which are used here and in many other papers on paging [3]: A paging algorithm is called *conservative* [24] if it incurs at most k page faults on any consecutive subsequence of the input containing k or fewer distinct page references. LRU and FIFO belong to this class. Similarly, a paging algorithm is called a *marking* [4] algorithm if for any k-phase, once a page has been requested in that phase, it is not evicted for the duration of that phase. LRU, FAR^G, and FWF are marking algorithms. All conservative and marking algorithms are k-competitive [4, 23].

If $\mathcal{A}$ is a paging algorithm, we let $\mathcal{A}(I)$ denote $\mathcal{A}$'s cost (number of faults) on the input (request) sequence I. We now adapt relative interval analysis from [13] to access graphs. Let $\mathcal{A}$ and $\mathcal{B}$ be two algorithms. We define the following notation:

$$\mathrm{Min}^G_{\mathcal{A},\mathcal{B}}(n) = \min_{|I|=n, I \in \mathcal{L}(G)} \{\mathcal{A}(I) - \mathcal{B}(I)\}$$

$$\mathrm{Max}^G_{\mathcal{A},\mathcal{B}}(n) = \max_{|I|=n, I \in \mathcal{L}(G)} \{\mathcal{A}(I) - \mathcal{B}(I)\}$$

$$\mathrm{Min}^G(\mathcal{A},\mathcal{B}) = \lim_{n\to\infty} \inf \frac{\mathrm{Min}^G_{\mathcal{A},\mathcal{B}}(n)}{n}$$

$$\mathrm{Max}^G(\mathcal{A},\mathcal{B}) = \lim_{n\to\infty} \sup \frac{\mathrm{Max}^G_{\mathcal{A},\mathcal{B}}(n)}{n}$$

Definition 3. The *relative interval* of two algorithms $\mathcal{A}$ and $\mathcal{B}$ with respect to the access graph, G, is

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

$\mathcal{B}$ is said to have *better performance than* $\mathcal{A}$ if $\mathrm{Max}^G(\mathcal{A},\mathcal{B}) > |\mathrm{Min}^G(\mathcal{A},\mathcal{B})|$.
$\mathcal{B}$ is said to *dominate* $\mathcal{A}$ if $\mathcal{I}^G(\mathcal{A},\mathcal{B}) = [0,\beta]$ for some $\beta > 0$. $\square$

Note that in the above, $\mathrm{Max}^G(\mathcal{A},\mathcal{B}) = -\mathrm{Min}^G(\mathcal{B},\mathcal{A})$.

This definition generalizes the one from [13] in that the original definition is the special case where G is the complete graph, which is the same as saying that there are no restrictions on the sequences. We omit G in the notation when G is complete, since this corresponds to the normal case without an access graph.

Note that if $\mathcal{B}$ *dominates* $\mathcal{A}$, this means that $\mathcal{A}$ does not outperform $\mathcal{B}$ on any sequence (asymptotically), while there are sequences on which $\mathcal{B}$ outperforms $\mathcal{A}$. Also, when $\mathrm{Max}^G(\mathcal{A},\mathcal{B})$ is close to 0, this indicates that $\mathcal{A}$'s performance is not much worse than that of $\mathcal{B}$'s.

As a remark, all the cost functions in this paper behave nicely, so taking the infimum or supremum is not necessary for the limits to exist. Thus, in all cases, one can just think about the limit.

One can debate whether or not it is intuitive that the algorithms sometimes appear as subscripts and sometimes as arguments (in parenthesis) in the above, but we have decided to stay as close as possible to the notation and terms introduced in [13], so that their notation is as above when the superscript G is removed. We have also kept their term "better performance than", but the reader should be aware that this is now a formal term, expressing the result of comparing algorithms using relative interval analysis, rather than a term referring to some real-life benchmark results. And of course there will be results where the measure states that "$\mathcal{A}$ has better performance than $\mathcal{B}$", but one could have another view on this in practice. This is also the situation for any other measure, including competitive analysis; see [14] for numerous examples of this.

The following general lemmas will prove helpful later. The first observation is well known for k-phases [3]:

Lemma 1. If a request sequence I contains either

- b complete k-phases, or
- b complete k-blocks defined with respect to any conservative or marking algorithm, $\mathcal{B}$,

then any algorithm, $\mathcal{A}$, has at least $b+k-1$ faults on I.

Proof Let p be the page requested first in Phase i of I and let I' be the subsequence starting with the second request in Phase i and ending right after the first request in Phase $i+1$. Since there are k different pages in I' different from p, and p is in cache right after it has been processed, any algorithm, $\mathcal{A}$, must fault at least once in I'. Thus, $\mathcal{A}$ must fault at least $k+1$ times on Phase 1 and the first request in Phase 2, and then at least once for the next $b-2$ k-phases, summing to $b+k-1$.

The only properties used in the above are the following: First, there are at least k distinct requests in a k-phase, and, second, for any phase, the first request is different from any request in the previous phase; specifically, the first requests in two subsequent k-phases are different. Any conservative or marking algorithm, $\mathcal{B}$, gives rise to such k-blocks. □

Lemma 2. Assume that for two algorithms $\mathcal{A}$ and $\mathcal{B}$ and an access graph G, there exist functions f and g such that

- for all $I \in \mathcal{L}(G)$, $\mathcal{A}(I) - \mathcal{B}(I) \leq f(b_I)$ and $|I| \geq g(b_I)$, where b_I denotes either the number of complete k-phases in I or the number of k-blocks in I defined with respect to a conservative or marking algorithm, and
- the limit $\lim_{b\to\infty} \frac{f(b)}{g(b)}$ exists.

Then $\mathrm{Max}^G(\mathcal{A},\mathcal{B}) \leq \lim_{b\to\infty} \frac{f(b)}{g(b)}$.

Proof In this proof, we will take the word "phase" to mean either a k-phase or a k-block.

We define a sequence of request sequences as follows. For $n \geq 1$, let I_n be a sequence of length n such that I_n maximizes $\mathcal{A}(I) - \mathcal{B}(I)$ over all sequences I of length n.

By construction,

$$\mathrm{Max}^{G}_{\mathcal{A},\mathcal{B}}(n) = \max_{|I|=n, I \in \mathcal{L}(G)} \{\mathcal{A}(I) - \mathcal{B}(I)\} = \mathcal{A}(I_n) - \mathcal{B}(I_n) \leq f(b_{I_n}),$$

and by assumption, $|I_n| \geq g(b_{I_n})$. Thus $\frac{\mathrm{Max}^{G}_{\mathcal{A},\mathcal{B}}(n)}{|I_n|} \leq \frac{f(b_{I_n})}{g(b_{I_n})}$. Now,

$$\mathrm{Max}^{G}(\mathcal{A},\mathcal{B}) \leq \limsup_{n \to \infty} \frac{f(b_{I_n})}{g(b_{I_n})} \leq \limsup_{b \to \infty} \frac{f(b)}{g(b)} = \lim_{b \to \infty} \frac{f(b)}{g(b)}.$$

The final equality holds since we have assumed that the limit exists. □

The proof of the following is analogous to the lemma just proven. Note, however, that the function f now has image in $\mathbb{R}^{-}$.

Lemma 3. Assume that for two algorithms $\mathcal{A}$ and $\mathcal{B}$ and an access graph G, there exist functions f and g such that

- for all $I \in \mathcal{L}(G)$, $\mathcal{A}(I) - \mathcal{B}(I) \geq f(b_I)$ and $|I| \geq g(b_I)$, where b_I denotes either the number of complete k-phases in I or the number of k-blocks in I defined with respect to a conservative or marking algorithm, and
- the limit $\lim_{b \to \infty} \frac{f(b)}{g(b)}$ exists.

Then $\mathrm{Min}^{G}(\mathcal{A},\mathcal{B}) \geq \lim_{b \to \infty} \frac{f(b)}{g(b)}$.

3. Complete Graphs

Having the complete graph as an access graph is equivalent to having no restrictions on the input. Thus, LRU and FIFO are equivalent on complete graphs under both competitive analysis and relative worst order analysis, since they are equivalent under these measures without considering access graphs. It also means that the results of this section hold in the original model for relative interval analysis [13]. In [13], it is shown that $[-\frac{k-1}{k}, \frac{k-1}{2k-1}] \subseteq \mathcal{I}(\mathrm{FIFO}, \mathrm{LRU})$. Below, we answer an open question from [13], proving that this is tight.

The argument for the Min value uses the following lemma, which is a generalization of [13, Theorem 3].

Lemma 4. Let $\mathcal{A}$ be a k-competitive algorithm and $\mathcal{B}$ be any paging algorithm, then for any access graph G,

$$\mathrm{Min}^G(\mathcal{B}, \mathcal{A}) \geq -1 + \frac{1}{k} \text{ and } \mathrm{Max}^G(\mathcal{A}, \mathcal{B}) \leq 1 - \frac{1}{k}.$$

Proof For each positive integer n, let I_n be a sequence of length n, respecting the access graph G, which maximizes $\mathcal{A}(I) - \mathcal{B}(I)$.

$$\mathrm{Max}^G_{\mathcal{A},\mathcal{B}}(n) = \max_{|I|=n, I \in \mathcal{L}(G)} \{\mathcal{A}(I) - \mathcal{B}(I)\} = \mathcal{A}(I_n) - \mathcal{B}(I_n) \leq \mathcal{A}(I_n) - \mathrm{OPT}(I_n).$$

Since $\mathcal{A}$ is k-competitive, there exists a constant c such that for all sequences, I, $\mathcal{A}(I) \leq k\,\mathrm{OPT}(I) + c$. In particular, for I_n, $\mathrm{OPT}(I_n) \geq \frac{\mathcal{A}(I_n)-c}{k}$. In addition, $\mathcal{A}$ cannot fault on more than n requests on I_n, so

$$\begin{aligned}
\mathrm{Max}^G(\mathcal{A}, \mathcal{B}) &\leq \limsup_{n\to\infty} \frac{\mathcal{A}(I_n) - \mathrm{OPT}(I_n)}{n} \leq \limsup_{n\to\infty} \frac{\mathcal{A}(I_n) - \frac{\mathcal{A}(I_n)-c}{k}}{n} \\
&= \limsup_{n\to\infty} \frac{\mathcal{A}(I_n) - \frac{\mathcal{A}(I_n)}{k}}{n} \leq \limsup_{n\to\infty} \frac{\mathcal{A}(I_n) - \frac{\mathcal{A}(I_n)}{k}}{\mathcal{A}(I_n)} \\
&= 1 - \frac{1}{k}
\end{aligned}$$

Since $\mathrm{Min}^G(\mathcal{B}, \mathcal{A}) = -\mathrm{Max}^G(\mathcal{A}, \mathcal{B})$, $\mathrm{Min}^G(\mathcal{B}, \mathcal{A}) \geq -1 + \frac{1}{k}$. □

Lemma 5. For any access graph G,

$$-1 + \frac{1}{k} \leq \mathrm{Min}^G(\mathrm{FIFO}, \mathrm{LRU}) \text{ and } \mathrm{Max}^G(\mathrm{FIFO}, \mathrm{LRU}) \leq \frac{1}{2} - \frac{1}{4k-2}.$$

Proof Since LRU has competitive ratio k, by Lemma 4, $\mathrm{Min}^G(\mathrm{FIFO}, \mathrm{LRU}) \geq -1 + \frac{1}{k}$.

We now consider the Max value. Given a request sequence I, we let B_i denote the ith k-block for FIFO. Assume that there are b complete k-blocks.

FIFO faults k times per complete k-block and up to $k-1$ times for the possible final k-block. Thus, $\text{FIFO}(I) \leq bk+(k-1)$. Assume that LRU faults α_i times in B_i. By Lemma 1, LRU faults at least $b+k-1$ times. Thus, $\Sigma_{i=1}^{b}\alpha_i \geq b+k-1$.

We now compute a lower bound on the length of the request sequence I based on the number of complete k-blocks in it and the behavior of the algorithms on it.

As a first step, with every request on which FIFO faults and LRU has a hit, we associate a unique request on which FIFO has a hit. Let r be such a request to a page p in B_i. Since it is a hit for LRU, p must have been requested in the maximal subsequence of requests I' consisting of k distinct pages and ending just before r. Consider the first such request, r', in I'. If it were a fault for FIFO, FIFO could not have faulted again on r. Thus, r' was a hit for FIFO and we associate r' with r.

To establish that the association is one-to-one, assume that r' also gets associated with a request r''. Without loss of generality, assume that r'' is later than r. For FIFO to fault on both r and r'', there must be at least k distinct pages different from p in between r and r''. However, since we are assuming that LRU has a hit on r'', by the property of LRU, the page requested by r'' must have been requested during the same k distinct pages. Thus, by the construction above, the request that gets associated with r'' (and r) will be later than r, which is a contradiction.

Thus, for each of the at least $k-\alpha_i$ requests in B_i where FIFO faults and LRU has hit, by the procedure above, we have identified an additional request where FIFO has a hit. In total, there are at least $\Sigma_{i=1}^{b}(k-\alpha_i) = bk - \Sigma_{i=1}^{b}\alpha_i$ hits for FIFO on distinct requests in I. Since there are b complete k-blocks, there are also at least bk faults. Thus, the length of I is at least $2bk - \Sigma_{i=1}^{b}\alpha_i$, and

$$\frac{\text{FIFO}(I)-\text{LRU}(I)}{|I|} \leq \frac{bk+k-1-\Sigma_{i=1}^{b}\alpha_i}{2bk-\Sigma_{i=1}^{b}\alpha_i}.$$

By the lower bound on $\Sigma_{i=1}^{b}\alpha_i$ above, and the arithmetic observation that $\frac{u-y}{v-y} <$

$\frac{u-x}{v-x}$, if $u < v$ and $x < y < v$, we have that

$$\frac{bk+k-1-\Sigma_{i=1}^{b}\alpha_i}{2bk-\Sigma_{i=1}^{b}\alpha_i} \leq \frac{bk+k-1-(b+k-1)}{2bk-(b+k-1)} = \frac{b(k-1)}{b(2k-1)-k+1}.$$

By Lemma 2, since $\lim_{b\to\infty} \frac{b(k-1)}{b(2k-1)-k+1} = \frac{k-1}{2k-1}$,

$$\mathrm{Max}^G(\mathrm{FIFO},\mathrm{LRU}) \leq \frac{k-1}{2k-1} = \frac{1}{2} - \frac{1}{4k-2}.$$

□

From [13] and Lemma 5, we have the following:

Theorem 1. $\mathcal{I}(\mathrm{FIFO},\mathrm{LRU}) = [-1+\frac{1}{k}, \frac{1}{2}-\frac{1}{4k-2}]$.

3.1. FWF

FWF performs very poorly compared to the other algorithms considered here. The following is folklore:

Lemma 6. For any sequence I and any conservative or marking algorithm $\mathcal{A}$, we have $\mathcal{A}(I) \leq \mathrm{FWF}(I)$.

This implies that for any access graph G and any request sequence I respecting G, $\mathcal{A}(I) \leq \mathrm{FWF}(I)$ and so

$$\mathrm{Min}^G(\mathrm{FWF},\mathrm{LRU}) = \mathrm{Min}^G(\mathrm{FWF},\mathrm{FIFO}) = \mathrm{Min}^G(\mathrm{FWF},\mathrm{FAR}^G) = 0.$$

Thus, LRU, FIFO, and FAR^G all dominate FWF.

The upper bound of $1-\frac{1}{k}$ from Lemma 4 is tight for FWF versus LRU for any access graph containing a path on $k+1$ vertices, and it is tight for FWF versus FAR^{P_N} on a path containing at least $k+1 \leq N$ vertices. Note that a cycle on $k+1$ vertices contains a path on $k+1$ vertices, but FAR^{C_N} does not behave identically to FAR^{P_N}. The following theorem regards path graphs, which are the topic of the next section, but it has implications already in this section.

Theorem 2. For the path access graph P_N, where $N \geq k+1$, and $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FAR}^{P_N}\}$,

$$\mathcal{I}^{P_N}[\mathrm{FWF}, \mathcal{A}] = \left[0, 1 - \frac{1}{k}\right].$$

For LRU, the results hold more generally for any graph containing P_{k+1}.

Proof Consider the sequence $I_n = \langle 1, 2, \ldots, k, k+1, k, \ldots, 2\rangle^n$, respecting P_N. Since LRU and FAR behave identically when the access graph is a path, we have

$$\mathrm{LRU}(I_n) = \mathrm{FAR}^{P_N}(I_n) = 2n + k - 1 \text{ and } \mathrm{FWF}(I_n) = 2kn.$$

Therefore,

$$\lim_{n\to\infty} \frac{\mathrm{FWF}(I_n) - \mathrm{LRU}(I_n)}{|I_n|} = \lim_{n\to\infty} \frac{\mathrm{FWF}(I_n) - \mathrm{FAR}^{P_N}(I_n)}{|I_n|} = \frac{k-1}{k}.$$

By Lemma 4, this gives

$$\mathrm{Max}^{P_N}(\mathrm{FWF}, \mathrm{LRU}) = \mathrm{Max}^{P_N}(\mathrm{FWF}, \mathrm{FAR}^{P_N}) = 1 - \frac{1}{k}.$$

Lemma 6 shows that LRU and FAR^{P_N} dominate FWF. □

As defined in the introduction, FAR^G operates relative to some access graph G and first uses distances in the access graph in order to decide which pages to evict, and in cases of ties, uses LRU as its tie-breaking strategy. Since the distance between any two vertices in a complete graph is one, FAR^G and LRU behave identically when G is complete. Thus, the above result also holds for both LRU and FAR for complete graphs containing at least $k+1$ vertices. Recall that we omit the superscript to FAR^G and write FAR exactly when G is complete.

This gives as a corollary the following, which also follows from [13, Theorem 2]:

Corollary 1. For $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FAR}\}$, we have $\mathcal{I}[\mathrm{FWF}, \mathcal{A}] = \left[0, 1 - \frac{1}{k}\right]$.

Except for a tiny discrepancy when k is even, this result also holds for FWF versus FIFO:

Theorem 3. For any graph G containing a path with $k+1$ vertices, if k is odd, then

$$\mathcal{I}^G[\mathrm{FWF}, \mathrm{FIFO}] = \left[0, 1 - \frac{1}{k}\right],$$

and if k is even, then

$$\left[0, 1 - \frac{1}{k} + \frac{1}{k^2}\right] \subseteq \mathcal{I}^G[\mathrm{FWF}, \mathrm{FIFO}] \subseteq \left[0, 1 - \frac{1}{k}\right].$$

Proof Let $h = \lfloor (k+1)/2 \rfloor$. Define the subsequence

$$S_i = \langle h+i, h+i-1, ..., h, ..., h-i, h-i+1, ..., h, ..., h+i \rangle$$

and define the subsequence R which starts with page h and continues with the requests $S_1, S_2, ..., S_{h-1}$. This initial part of every sequence in our family of sequences ensures that the order in which FIFO faults on requests is always $\langle h, h+1, h-1, h+2, h-2, ..., 2h-1, 1 \rangle$. The value $2h-1$ is k if k is odd and $k-1$ if k is even.

Suppose k is odd. Let $I_n = \langle R, K_n \rangle$, where $J = \langle k+1, k, ..., 1, 2, ...k \rangle^h$ and $K_n = J^n$. FWF and FIFO both fault $2h-1$ times on R. FWF faults $2khn$ times on K_n. On the first request to page $k+1$ in I_n, FIFO evicts page h. Thus, after the fault on the request to $k+1$, its only fault while going "left" (towards lower page numbers) for the first time in subsequence J is on page h, and its only fault going "right" is on page $h+1$. On the ith iteration ($i \leq h-1$) of subsequence J, it faults on the request to $h-i+1$ going left and on the request to $h+i$ going right. On iteration h, it only faults on the request to page 1, so FIFO has the same cache configuration immediately after having processed subsequence J as it had immediately before. Thus, FIFO has $k+1$ faults on subsequence J, giving $(k+1)n$ in all. The number of requests in K_n is $2khn$. Thus, $\lim_{n\to\infty} \frac{\mathrm{FWF}(I_n)-\mathrm{FIFO}(I_n)}{|I_n|} = \frac{2khn-(k+1)n}{2khn} = 1 - \frac{1}{k}$, using that $2h-1=k$.

Suppose k is even. We define similar sequences, but let $I_n = \langle R, k, K_n \rangle$, since page k is not requested yet. FIFO will still fault $k+1$ times on subsequence J, but now $2h-1 = k-1$, so

$$\lim_{n\to\infty} \frac{\mathrm{FWF}(I_n) - \mathrm{FIFO}(I_n)}{|I_n|} = \frac{2khn - (k+1)n}{2khn} = 1 - \frac{k+1}{k^2}.$$

Lemma 6 shows that FIFO dominates FWF. □

4. Path Graphs

In this section, we analyze path access graphs, P_N, with N vertices. As in the other sections, we assume that $N \geq k+1$.

By Theorem 1, for any access graph G, $\mathrm{Max}^G(\mathrm{FIFO}, \mathrm{LRU}) \leq \frac{1}{2} - \frac{1}{4k-2}$. For paths, the value is slightly smaller. We obtain a tight result, establishing that $\mathrm{Max}^{P_N}(\mathrm{FIFO}, \mathrm{LRU}) = \frac{1}{2} - \frac{1}{2k}$, where the upper bound is the harder result.

Lemma 7. For the path access graph P_N,

$$\mathrm{Max}^{P_N}(\mathrm{FIFO}, \mathrm{LRU}) \leq \frac{1}{2} - \frac{1}{2k}.$$

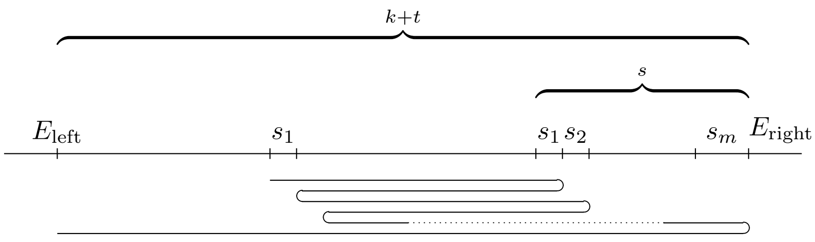
Proof Consider any request sequence I. We divide the sequence up into phases as follows (these are *not* k-phases). Initially, define a direction by where LRU makes its kth fault compared with its cache content. To be precise, note that LRU's cache content forms a connected subpath. When it faults, it is on a page immediately next to one of the ends of this subpath. Without loss of generality, we assume that the path is horizontal and that the fault happens going to the right on the path. Left and right are now well defined, and we will use the term "fault to the right" (similar for left) to mean that LRU faults on the page immediately next to the rightmost page on the path which is currently in LRU's cache.

We start the first phase with the first request and later explain how subsequent phases are started. In all the phases, we start to the left. As usual, the first phase is different from the rest. In the first phase, we get, by definition, $k-1$ faults, after which we get the kth fault going to the right.

In all phases, except the first, LRU has the first $k-1$ distinct pages that will be requested during that phase in cache. In all phases, the first fault by LRU in the phase, after having processed the first $k-1$ distinct pages, is to the right. We maintain this as an invariant that holds at the start of any phase.

The exception in the first phase, adding an extra $k-1$ faults to the cost of LRU as compared with the analysis below of all other phases, will not influence the result in the limit for the length of the request sequence going towards infinity.

We want to analyze a phase where LRU faults to the right before it faults to the left again. These faults to the right may not appear consecutively. There may be some faults in a row, but then there may be hits and then faults again, etc. Thus, assume that there are m maximal subsequences of requests to the right where LRU faults—all of this before LRU faults going to the left again. Assume further that these maximal subsequences of requests give rise to $s_1, s_2, \ldots, s_m$ faults, respectively, where, by definition, $m \geq 1$, and let $s = \Sigma_{i=1}^{m} s_i$.



For now, we assume that for all i, $s_i < k$. Thus, LRU moves left and right at least m times; maybe more times where it does not give rise to faults. Since it does not fault going to the left during these turns, the faults are to pages further and further to the right. Let E_{right} denote the extreme rightmost position it reaches during these faults to the right.

When LRU faults again to the left after having processed E_{right}, we consider the leftmost node E_{left}, where LRU faults after the s faults described above, but before it faults to the right again. We end the phase with the first request to E_{left} after the s faults. We define subsequent phases inductively in the same way, starting with the first request not included in the previous phase, possibly leaving an incomplete phase at the end.

We now consider the costs of the algorithms and the length of the sequence per phase. LRU faults s times going to the right during the m turns in the phase. Additionally, LRU must fault at least t times going from E_{right} to E_{left}, where t is defined by there being $k+t$ nodes between E_{left} and E_{right}, including both endpoints. This sums up to at least $s+t$ faults.

For FIFO, we postpone the discussion of the first s_1 distinct pages seen in a phase. Just to avoid any confusion, note that these pages are immediately to the right of E_{left} (the endpoint of the previous phase) and thus not the pages that LRU faults on. After that, consider the maximal subsequence of at most k distinct pages. This subsequence starts with the (s_1+1)th distinct request (the last request to it before the s_2 faults) and continues up to, but not including the first request that LRU has one of its s_2 faults on. We know that there are at most k pages there, because LRU only faults s_1 times there. Assume that FIFO faults f_1 times on this subsequence. Since FIFO is conservative, $f_1 \leq k$.

We define more such subsequences repeatedly, the $(m-1)$st of these ending just before LRU's first fault of the s_m faults, and the mth including the s_m faults and k of the $k+t$ nodes before we reach E_{left}. Finally, we return to the question of the first s_1 distinct pages seen in the phase. These overlap with the "t pages" from the previous phase; otherwise we would not have started the phase where we did. If FIFO faults on one of these pages when going through the t pages in the previous phase, it will not fault on them again in this phase. Thus, we only have to count them in one phase, and choose to do this in the previous phase. In total, FIFO faults at most $(\Sigma_{i=1}^{m} f_i)+t$ times, and for all i, $f_i \leq k$.

The difference between the cost of FIFO and LRU is then at most $(\Sigma_{i=1}^{m} f_i)+t-(s+t) = (\Sigma_{i=1}^{m} f_i)-s = (\Sigma_{i=1}^{m}(f_i-1))-(s-m)$.

From the analysis of FIFO above, knowing that on a subsequence of length at most k, FIFO can fault at most once on any given page, if it faults f_i times, the subsequence has at least f_i distinct pages. Given that the subsequence starts at the left end of the "s_i pages" and ends at the right end of the "s_i pages", all pages that FIFO faults on, except possibly the leftmost, must be requested at least twice, giving at least $2f_i-1$ requests. So, the length of the sequence is at least $(\Sigma_{i=1}^{m}(2f_i-1))+t$.

We now equip each variable with a superscript denoting the phase number, letting m^j denote the number of maximal subsequences of requests to the right where LRU faults in the jth phase, letting f_i^j denote the number of faults for the ith of these maximal sequences in the jth phase, and letting t^j denote the number of pages from E_{left} to E_{right} in the jth phase minus k, i.e., $E_{\text{right}} - E_{\text{left}} + 1 - k$. We now sum up over all phases to establish a lower bound on the length of the request sequence.

First, the total length, L, is at least

$$L \geq \Sigma_j(\Sigma_{i=1}^{m^j}(2f_i^j - 1)) + t^j = \Sigma_j(\Sigma_{i=1}^{m^j} 2f_i^j) - m^j + t^j.$$

Since s expresses how far we move to the right and t how far we move to the left, and the whole path has a bounded number of nodes N, we have that $\Sigma_j t^j \geq \Sigma_j s^j - N$. Thus, $L \geq (\Sigma_j(\Sigma_{i=1}^{m^j} 2f_i^j) - m^j + s^j) - N$.

I has a number of complete phases and then some extra requests in addition to that. There must exist a fixed constant c independent of I such that the cost of FIFO on the extra part of any sequence is bounded by c. This follows since there is a limit of N on how far requests can move to the right. So if requests never again come so far to the left that LRU faults, all requests thereafter are to only k pages. This added constant can also take care of the initial extra cost of $k - 1$. Since we are just using a lower bound on the sequence length, we can ignore the length of a possibly incomplete phase at the end. Thus,

$$\begin{aligned}
\frac{\text{FIFO}(I) - \text{LRU}(I)}{|I|} &\leq \frac{c + \Sigma_j \Sigma_{i=1}^{m^j}(f_i^j - 1) - (s^j - m^j)}{-N + \Sigma_j(\Sigma_{i=1}^{m^j} 2f_i^j) - m^j + s^j} \\
&\leq \frac{c + \Sigma_j \Sigma_{i=1}^{m^j}(f_i^j - 1)}{-N + \Sigma_j \Sigma_{i=1}^{m^j} 2f_i^j} \\
&\leq \frac{c + \Sigma_j m^j(k-1)}{-N + \Sigma_j m^j 2k} \\
&= \frac{c + (k-1)\Sigma_j m^j}{-N + 2k\Sigma_j m^j}
\end{aligned}$$

The second inequality follows since $s^j \geq m^j$, and the third inequality follows because $\frac{f_i^j - 1}{2f_i^j} \leq \frac{1}{2}$ and $k \geq f_i$ implies that $\frac{f_i^j - 1}{2f_i^j} \leq \frac{k-1}{2k}$.

For sequences where the number of phases does not approach infinity, as argued above, FIFO's cost will be bounded. For the number of phases approaching infinity, $\lim_{j\to\infty} \frac{c+(k-1)\Sigma_j m^j}{-N+2k\Sigma_j m^j} = \frac{k-1}{2k} = \frac{1}{2} - \frac{1}{2k}$, which implies the result.

Now, for this proof, we assumed that $s_i < k$. If $s_i \geq k$, we simply terminate the phase after the processing of the s_i requests that LRU faults on, and continue to define phases inductively from there. All the bounds from above hold with $t = 0$ and the observation that FIFO will not fault on the first s_1 requests in the next phase. The direction of the construction is now reversed. In this process, whenever we reverse the direction as above, we also rename the variable s to t and t to s, such that s continues to keep track of movement to the right and t of movement to the left, and the inequality $\Sigma_j t^j \geq \Sigma_j s^j - N$ still holds. □

Lemma 8. For the path access graph P_N,

$$\text{Max}^{P_N}(\text{FIFO}, \text{LRU}) = \frac{1}{2} - \frac{1}{2k}.$$

Proof The upper bound was shown in Lemma 7. Consider the family of sequences $I_n = \langle 1, 2, \ldots, k, k+1, k, k-1, \ldots, 2\rangle^n$. In each iteration, except the first, LRU faults twice (on pages 1 and $k+1$), whereas FIFO faults on pages 1 through $k+1$ in every iteration. So on this family, $\lim_{n\to\infty} \frac{\text{FIFO}(I_n)-\text{LRU}(I_n)}{|I_n|} = \frac{k-1}{2k} = \frac{1}{2} - \frac{1}{2k}$, and the maximum must be at least that large. □

Since LRU is optimal on paths, this gives :

Theorem 4. $\mathcal{I}^{P_N}[\text{FIFO}, \text{LRU}] = [0, \frac{1}{2} - \frac{1}{2k}]$, and LRU dominates FIFO on paths.

Note that FAR^{P_N} and LRU perform identically on paths, P_N, so FAR^{P_N} also dominates FIFO with the same interval.

5. Star Graphs

We let S_N denote a star graph with N vertices. A star graph is a tree, with a central vertex, s, which is adjacent to $N-1$ leaves. We again assume that $N \geq k+1$.

Lemma 9. If a request sequence I respecting the star access graph S_N contains b complete k-phases, then FIFO incurs at least $b + \left\lfloor \frac{b-3}{k} \right\rfloor + k$ faults on I. For $k > 1$, $\mathrm{FIFO}(I) \geq b + \frac{b}{k} + k - 2$.

Proof This proof is similar to the one presented for Lemma 1. The main difference is that with FIFO's eviction strategy, the central vertex of S_N, denoted by s, will sometimes be evicted and, thus, give rise to more faults during the course of FIFO serving I. We lower bound the number of times this happens as follows.

Since I respects S_N, every alternate request in I is for the central vertex s. Thus, one of the first two requests in I is for s. These two possibilities give rise to two cases, which we will analyze separately. As explained earlier, the first phase and the first request of the second phase together give rise to $k+1$ faults.

If s is the first page requested in I, then FIFO will evict s when serving the first request in Phase 2. Since the next page requested is s, FIFO will incur fault number $k+2$ at that request. After that fault, s is the most recent page to be added to FIFO's cache and, thus, will be evicted after k faults. Since FIFO must fault at least once in each phase, it must fault on s in phase $k+2$ at the latest. There are $b-2$ complete phases after the second one, and this pattern will repeat itself for the rest of the sequence. Thus, the number of faults incurred by FIFO is at least $k + 2 + b - 2 + \left\lfloor \frac{b-2}{k} \right\rfloor = b + k + \left\lfloor \frac{b-2}{k} \right\rfloor$.

If s was the second page to be requested in I, then fault number $k+2$ will lead to the eviction of s by FIFO. This will occur no later than the start of the third phase of I. Thus, FIFO's fault number $k+3$ will be due to s, and, counting as above, FIFO must fault at least $k + 3 + b - 3 + \left\lfloor \frac{b-3}{k} \right\rfloor = b + k + \left\lfloor \frac{b-3}{k} \right\rfloor$ times on I. Therefore, for either case, we have shown that $\mathrm{FIFO}(I) \geq b + \left\lfloor \frac{b-3}{k} \right\rfloor + k$. □

Lemma 10. For the star access graph S_N,

$$\mathrm{Min}^{S_N}(\mathrm{FIFO}, \mathrm{LRU}) = -\frac{1}{2} + \frac{1}{2(k-1)} + \frac{1}{2k(k-1)}$$

Proof Consider an arbitrary sequence I respecting the star access graph, and consider its division into k-phases. Since the central vertex occurs after each request to a leaf, each k-phase, except the last, must contain requests to $k-1$ different leaves, and must be of length at least $2(k-1)$. Suppose a sequence I has b k-phases, not counting the first empty phase. Then, there are at least $b-1$ complete phases, so $|I| \geq 2(k-1)(b-1)+1$, and, by Lemma 9, since $k \geq 2$, $\text{FIFO}(I) \geq b + \frac{b}{k} + k - 3$. LRU faults only on the leaves and at most once in each phase, so $\text{LRU}(I) \leq (k-1)(b-1)+k$. Thus, $\text{FIFO}(I) - \text{LRU}(I) \geq b + \frac{b}{k} - 3 - (k-1)(b-1) = -b(k-1) + b + \frac{b}{k} + (k-4)$. By Lemma 3, $\text{Min}^{S_N}(\text{FIFO}, \text{LRU}) \geq -\frac{1}{2} + \frac{1}{2(k-1)} + \frac{1}{2k(k-1)}$.

We show that this bound is tight by analyzing the following sequence.

$$\begin{aligned} I_n &= \langle P, J^n \rangle,\ J = \ B_1, \ldots, B_{k-1} \\ P &= \langle 1, s, 2, s, \ldots s, k-2, s, k-1, s, k-2, s, \ldots s, 2, s, 1, s \rangle \\ B_i &= \langle k, s, k-1, s, \ldots, s, 1, s \rangle, \text{ for } 1 \leq i \leq k-1 \end{aligned}$$

We note that page k does not appear in subsequence P and that all the B_i subsequences are identical (we use the index for reference). Since $|B_i| = 2k$, the sequence length is $|I_n| = 2(2k-3) + 2k(k-1)n$. LRU starts sequence B_1 with a fault on the request to page k, thereby evicting page $k-1$. LRU then faults on the request to page $k-1$ and evicts page $k-2$. This repeats and ends with the eviction of page k at the request to page 1 such that page $k-1$ is the least recently used page. Thus, LRU faults everywhere except on the central vertex s, which is never evicted by LRU. Since LRU's cache configuration—content as well as the relative ordering of the recency of pages—is the same at the end of subsequence B_1 as it was at the end of subsequence P, the same pattern must be repeated in each subsequence B_i. Thus, $\text{LRU}(I_n) = k + (k-1)kn$.

FIFO has three faults in subsequence B_1: On the request to page k, where page 1 is evicted, and at the last two requests of subsequence B_1. So FIFO ends subsequence B_1 with page 2 being outside its cache. From there onwards, FIFO faults exactly once in each of the subsequences B_i, $2 \leq i \leq k-1$, at the request to page i, on which it evicts page $i+1$. Therefore, FIFO ends each subsequence J with page k outside its cache and, hence, the above described fault and eviction pattern is repeated in every subsequence J. This gives the cost $\mathrm{FIFO}(I_n) = k + (k+1)n$, and $\lim_{n\to\infty} \frac{\mathrm{FIFO}(I_n) - \mathrm{LRU}(I_n)}{|I_n|}$ equals

$$\lim_{n\to\infty} \frac{k + (k+1)n - (k + (k-1)kn)}{2(2k-3) + 2k(k-1)n} = -\frac{1}{2} + \frac{k+1}{2k(k-1)}$$

Thus, $\mathrm{Min}^{S_N}(\mathrm{FIFO}, \mathrm{LRU}) \leq -\frac{1}{2} + \frac{k+1}{2k(k-1)} = -\frac{1}{2} + \frac{1}{2(k-1)} + \frac{1}{2k(k-1)}$, and equality holds. $\square$

Lemma 11. For the star access graph S_N,

$$\mathrm{Max}^{S_N}(\mathrm{FIFO}, \mathrm{LRU}) = \frac{1}{4} + \frac{1}{8k-12}.$$

Proof We give a sequence respecting S_N for $N \geq k+1$ giving rise to the stated ratio. Let

$$I_n = \langle P, B^n \rangle, \text{ where } P = \langle 1, s, 2, s, \ldots, s, k-2, s, k-1, s \rangle \text{ and } B \text{ is}$$

$$\left[\begin{array}{cccccccccccccccc}
k-2, & s, & \dots\ s, & 2, & s, & 1, & s, & \mathbf{k}, & s, & 1, & s, & 2, & s, & \dots\ s, & k-2, & s \\
k-3, & s, & \dots\ s, & 1, & s, & k, & s, & \mathbf{k-1}, & s, & k, & s, & 1, & s, & \dots\ s, & k-3, & s \\
k-4, & s, & \dots\ s, & k, & s, & k-1, & s, & \mathbf{k-2}, & s, & k-1, & s & k, & s, & \dots\ s, & k-4, & s \\
\vdots & \vdots & \dots\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots\ \vdots & \vdots & \vdots \\
k, & s, & \dots\ s, & 4, & s, & 3, & s, & \mathbf{2}, & s, & 3, & s, & 4, & s, & \dots\ s, & k, & s \\
k-1, & s, & \dots\ s, & 3, & s, & 2, & s, & \mathbf{1}, & s, & 2, & s, & 3, & s, & \dots\ s, & k-1, & s
\end{array}\right]$$

Writing the sequence B like this is just to give an overview. The sequence is the concatenation of all the rows from top to bottom.

The column in bold indicates the requests that are faults for LRU. LRU faults on exactly one request in every row and so we have $\mathrm{LRU}(I_n) = k + kn$. FIFO faults on k distinct pages in each row, starting with the request at which LRU faults. Thus, $\mathrm{FIFO}(I_n) = k + k^2 n$. Furthermore, $|I_n| = 2(k-1) + (4k-6)kn$. Since

$$\lim_{n\to\infty} \frac{\mathrm{FIFO}(I_n) - \mathrm{LRU}(I_n)}{|I_n|} = \lim_{n\to\infty} \frac{k + k^2 n - (k + kn)}{2(k-1) + (4k-6)kn} = \frac{k-1}{4k-6},$$

we have that $\mathrm{Max}^{S_N}(\mathrm{FIFO}, \mathrm{LRU}) \geq \frac{k-1}{4k-6} = \frac{1}{4} + \frac{1}{8k-12}$.

To prove a tight upper bound on $\mathrm{Max}^{S_N}(\mathrm{FIFO}, \mathrm{LRU})$, we consider an arbitrary sequence I. We can assume without loss of generality that I does not contain any consecutive requests to the same page. This is because they give rise to hits for both algorithms and they do not change the future behavior of any of the algorithms. Thus, their difference in cost is unaffected. At the same time, repeated requests can only increase the length of the sequence, making the ratio lower.

We view sequence I as a partition of k-blocks with respect to FIFO, denoted by $B_1, \ldots, B_n$, ignoring the first empty block. Since both FIFO are LRU are conservative, each block, excluding perhaps the last one, must have requests to at least k distinct pages. The access graph is a star, so each request must be followed by a request to page s. The number of faults incurred by LRU in block B_i is denoted by α_i, where $\alpha_1 = k$. From the maximality of the blocks B_i, each block must have at least one fault for LRU. Since page s is never evicted from the cache by LRU, we have $1 \leq \alpha_i \leq k-1$, for $i > 1$.

We now find a lower bound on the length of the sequence B_i, first establishing a certain number of hits by FIFO. Consider a leaf request r that is a fault for FIFO, but a hit for LRU. Since it is not a fault for LRU, there must have been a request r' to the same page in the last $k-1$ distinct page requests. If r' were a fault for FIFO, then r would have to be a hit. Since it is not, r' must be a hit for FIFO. By definition, LRU incurs α_i faults on the sequence B_i. Recall that there are at least $k-1$ leaf requests where FIFO faults. Since it cannot fault twice on the same page in a k-block, these $k-1$ faults are on distinct pages. Thus, there are at least $k-1-\alpha_i$ distinct leaf requests where LRU has a hit while FIFO faults. As just argued, for each such request r, we can identify a request r' where FIFO has a hit, ensuring at least $k-1-\alpha_i$ unique hits for FIFO. The uniqueness follows from the fact that even though the hit we establish for FIFO could be in the previous block, B_{i-1}, it cannot be counted twice, since there are no more faults on that page after r' in B_{i-1}. Thus, if a hit r' for FIFO in B_{i-1} was identified from a request r in B_i, where FIFO faulted, then there is no such fault for FIFO in B_{i-1} from which we identify r' as a hit we can count.

The faults and the hits, together with the requests to s following each of them, gives us at least $2(k-1)+2(k-1-\alpha_i)$ requests. Since the terms not involving n disappear in the limit, the costs as well as the length of the first block, as well as a possibly incomplete final block, will not affect the result, and we leave them out in the fraction below.

$$\text{Max}^{S_N}(\text{FIFO}, \text{LRU}) \leq \max_{\substack{\alpha_2,\ldots,\alpha_n \\ \alpha_i \geq 1}} \left\{ \frac{\sum_{i=2}^{n} k - \alpha_i}{\sum_{i=2}^{n-1}(4k-4-2\alpha_i)} \right\}$$

This is maximized for $\alpha_i = 1$ for $2 \leq i \leq n$. Hence, $\text{Max}^{S_N}(\text{FIFO}, \text{LRU}) \leq \frac{k-1}{4k-6}$.

□

The algorithms FAR^{S_N} and LRU behave identically on star graphs, S_N. Neither of them ever evicts the central vertex. We state the result for both LRU and FAR^{S_N} in the main theorem, though FAR^{S_N} is not directly mentioned in the lemmas and proofs.

Theorem 5. For the star access graph S_N and $\mathcal{A} \in \{\text{LRU}, \text{FAR}^{S_N}\}$,

$$\mathcal{I}^{S_N}[\text{FIFO}, \mathcal{A}] = \left[-\frac{1}{2} + \frac{1}{2(k-1)} + \frac{1}{2k(k-1)}, \frac{1}{4} + \frac{1}{8k-12} \right]$$

Proof This follows directly from Lemmas 10 and 11. □

In [13], it was shown that $\text{Max}(\text{FIFO}, \text{LRU}) \geq \frac{k-1}{2k-1} = \frac{1}{2} - \frac{1}{4k-2}$. The above result shows that for star access graphs, that bound can be decreased by a factor of approximately two.

Since LRU and FAR^{S_N} perform identically on stars, S_N,

$$\text{Min}^{S_N}(\text{FAR}^{S_N}, \text{LRU}) = \text{Max}^{S_N}(\text{FAR}^{S_N}, \text{LRU}) = 0.$$

The star access graph is another example of where FWF performs poorly compared with the other algorithms.

Lemma 12. For the star access graph S_N, and any algorithm $\mathcal{B}$,

$$\text{Max}^{S_N}(\text{FWF}, \mathcal{B}) \leq \frac{1}{2}.$$

Furthermore, $\text{Max}^{S_N}(\text{FWF}, \text{FIFO}) \leq \frac{1}{2} - \frac{1}{2k(k-1)}$.

Proof Given any sequence I in S_N, it can be viewed as a partition of k-phases. Since it is a star, each complete phase must be of length at least $2(k-1)$, and, by Lemma 1, $\mathcal{B}$ must incur at least one fault for each phase. Since FWF can incur at most k faults in each phase, if there are n complete phases in I, then by Lemma 2, $\frac{\mathrm{FWF}(I)-\mathcal{B}(I)}{|I|} \leq \frac{n(k-1)}{2n(k-1)} = \frac{1}{2}$. Hence, $\mathrm{Max}^{S_N}(\mathrm{FWF}, \mathcal{B}) \leq \frac{1}{2}$.

For FIFO, by Lemma 9, $\mathrm{FIFO}(I) \geq n + \frac{n}{k} + k - 2$. From the upper bound on FWF, we get that $\mathrm{FWF}(I) \leq nk + k - 1$, we have $\frac{\mathrm{FWF}(I)-\mathrm{FIFO}(I)}{|I|} \leq \frac{n(k-1)-\left(\frac{n}{k}+1\right)}{2n(k-1)}$. By Lemma 2, $\mathrm{Max}^{S_N}(\mathrm{FWF}, \mathrm{FIFO}) \leq \frac{1}{2} - \frac{1}{2k(k-1)}$. □

Theorem 6. For the star access graph S_N, and $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FAR}^{S_N}\}$,

$$\mathcal{I}^{S_N}[\mathrm{FWF}, \mathcal{A}] = \left[0, \frac{1}{2}\right] \text{ and } \mathcal{I}^{S_N}[\mathrm{FWF}, \mathrm{FIFO}] = \left[0, \frac{1}{2} - \frac{1}{2k(k-1)}\right].$$

Proof By Lemma 6,

$$\mathrm{Min}^{S_N}(\mathrm{FWF}, \mathrm{LRU}) = \mathrm{Min}^{S_N}(\mathrm{FWF}, \mathrm{FIFO}) = \mathrm{Min}^{S_N}(\mathrm{FWF}, \mathrm{FAR}^{S_N}) = 0.$$

Consider the sequence $I_n = \langle P, (B_1, B_2)^n \rangle$, where $P = \langle 1, s, 2, s, \ldots, s, k-2, s, k-1, s \rangle$,

$$B_1 = \langle k, s, k-1, s, \ldots, s, 2, s \rangle, \text{ and } B_2 = \langle 1, s, 2, s, \ldots, s, k-1, s \rangle$$

Subsequences B_1 and B_2 have requests to k distinct pages, excluding the pages 1 and k, respectively.

LRU faults on the first request in each subsequence B_i. FWF flushes its cache at the start of each subsequence B_i. So $|I_n| = 4(k-1)n + 2(k-1)$, $\mathrm{LRU}(I_n) = 2n + k$, and $\mathrm{FWF}(I_n) = 2kn + k$. So $\lim_{n\to\infty} \frac{\mathrm{FWF}^{S_N}(I_n)-\mathrm{LRU}(I_n)}{|I_n|} = \frac{2(k-1)}{4(k-1)} = \frac{1}{2}$ and $\mathrm{Max}^{S_N}(\mathrm{FWF}, \mathrm{LRU}) \geq \frac{1}{2}$.

Let $I_n = \langle P, B^n \rangle$, where $P = \langle 1, s, 2, s, \ldots, s, k-2, s, k-1, s \rangle$ and

$$B = \begin{bmatrix} k & s & k-1 & \cdots & \cdots & 2 & s \\ 1 & s & k & \cdots & \cdots & 3 & s \\ 2 & s & 1 & \cdots & \cdots & 4 & s \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k-2 & s & k-3 & \cdots & \cdots & k & s \\ k-1 & s & k-2 & \cdots & \cdots & 1 & s \end{bmatrix}$$

The ith row is i-free. Hence, each row is of length $2(k-1)$ and $|I_n| = 2(k-1) + 2(k-1)kn$. Since FIFO only faults on the first request in each row, except the second row where it faults twice (on 1 and s), $\text{FIFO}(B) = k+1$ and $\text{FIFO}(I_n) = (k+1)n+k$. Since FWF flushes its cache at the start of each row, it incurs k faults in each row. Therefore, $\text{FWF}(B) = k^2$ and $\text{FWF}(I_n) = k^2n + k$. Thus, $\lim_{n\to\infty} \frac{\text{FWF}(I_n)-\text{FIFO}(I_n)}{|I_n|} = \frac{k(k-1)-1}{2(k-1)k} = \frac{1}{2} - \frac{1}{2k(k-1)}$.

Since FAR^{S_N} and LRU behave identically on S_N, by Lemma 12, we have proven all three identities. □

6. Cycle Graphs

We consider graphs consisting of exactly one cycle, containing $N > k$ vertices. We assume that $N > k$, since otherwise all pages of the cycle fit in cache. Thus, none of the algorithms will have to ever evict a page, and results become trivial.

For this section, it is convenient to work modulo N when indexing pages on the cycle. Thus, if $p < 1$ or $p > N$, we let p denote the page $((p-1) \mod N)+1$. We will not mention this again later in the proofs to follow.

It is easy to see that, according to relative interval analysis, FWF performs as poorly compared to LRU on cycles as it does on complete graphs.

Theorem 7. For the cycle access graph C_N,

$$\mathcal{I}^{C_N}[\text{FWF}, \text{LRU}] = \left[0, 1 - \frac{1}{k}\right]$$

Proof The sequence, $I_n = \langle 1, 2, \ldots, k, k+1, k, \ldots, 2\rangle^n$, respecting C_N, gives the right endpoint in conjunction with Lemma 4. The left endpoint is given by Lemma 6. □

In addition, these exist sequences respecting the cycle graph, where FIFO can perform as poorly compared to LRU as it can on complete graphs. Recall that for finite paths, though LRU is optimal, it cannot gain quite this significant an advantage over FIFO.

Lemma 13. For the cycle access graph C_N,

$$\mathrm{Max}^{C_N}(\mathrm{FIFO}, \mathrm{LRU}) \geq \frac{1}{2} - \frac{1}{4k-2}$$

Proof Let $I_n = \langle S_0, S_1, ..., S_n \rangle$, where

$$S_i = \langle i+k, i+k-1, \ldots, i+2, i+1, i+2, \ldots, i+k-1, i+k \rangle.$$

Clearly, $\mathrm{FIFO}(S_0) = \mathrm{LRU}(S_0) = k$.

In processing S_1, LRU only faults on the request to $1+k$, where it evicts page 1, which is not requested in S_1. In general, LRU faults only on the first request in each sequence S_i, evicting page i, which is not requested in S_i. Hence, $\mathrm{LRU}(I_n) = k+n$.

FIFO faults on the first request in S_1, evicting k, which is requested next. At that request page $k-1$ is evicted, leading to a fault on the following request, etc. In total, FIFO faults k times on S_1 and pages are brought into cache in the ordering $i+k$ through $i+1$. Thus, in general, when the processing of S_{i+1} starts, the situation repeats. Hence, we have $\mathrm{FIFO}(I_n) = k+kn$. The length of the sequence is $|I_n| = (2k-1)(n+1)$. So,

$$\begin{aligned}\mathrm{Max}^{C_N}(\mathrm{FIFO}, \mathrm{LRU}) &\geq \lim_{n\to\infty} \frac{\mathrm{FIFO}(I_n) - \mathrm{LRU}(I_n)}{|I_n|} \\ &= \lim_{n\to\infty} \frac{k+kn-(k+n)}{(2k-1)(n+1)} \\ &= \frac{k-1}{2k-1} = \frac{1}{2} - \frac{1}{4k-2}\end{aligned}$$

□

The following sequences were used in [13, Theorem 7] to show that $[-1 + \frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}] \subseteq \mathcal{I}[\mathrm{FIFO}, \mathrm{LRU}]$.

$$I_m = \langle P, B^m \rangle, \text{ where } P = \langle 1, 2, \ldots, k-1, k \rangle, \text{ and } B \text{ is}$$

$$\begin{bmatrix} k-1 & k-2 & \cdots & 2 & 1 & \mathbf{k+1} & 1 & 2 & \cdots & k-1 \\ k-2 & k-3 & \cdots & 1 & k+1 & \mathbf{k} & k+1 & 1 & \cdots & k-2 \\ k-3 & k-4 & \cdots & k+1 & k & \mathbf{k-1} & k & k+1 & \cdots & k-3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k & k-1 & \cdots & 3 & 2 & \mathbf{1} & 2 & 3 & \cdots & k \end{bmatrix}$$

$I_M = \langle P, B^M \rangle$, where $P = \langle 1, 2, \ldots, k-1, k, k-1, \ldots, 1 \rangle$, and

$$B = \begin{bmatrix} \mathbf{k+1} & k & k-1 & \cdots & 3 & 2 \\ \mathbf{1} & k+1 & k & \cdots & 4 & 3 \\ \mathbf{2} & 1 & k+1 & \cdots & 5 & 4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{k-1} & k-2 & k-3 & \cdots & k & k+1 \\ \mathbf{k} & k-1 & k-2 & \cdots & 2 & 1 \end{bmatrix}$$

These sequences respect C_{k+1}, the cycle access graph on $k+1$ vertices. Hence, that bound is applicable to cycles of length $k+1$ as well.

Proposition 1. For the cycle access graph C_{k+1},

$$\mathcal{I}^{C_{k+1}}[\text{FIFO}, \text{LRU}] = [-1 + \frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}].$$

Proof This follows from the results in [13], using the sequences above which respect the cycle, and Lemma 5. □

We now generalize these results to values of $N = k + r$, where $1 \leq r \leq k-1$. Note that these results become uninteresting if $r \geq k$, since the bound then does not imply that there are sequences where FIFO does better than the other algorithm.

Lemma 14. For the cycle access graph C_N, where $N = k + r$, $1 \leq r < k$,

$$\text{Min}^{C_N}(\text{FIFO}, \text{LRU}) \leq -1 + \tfrac{r}{k} \text{ and } \text{Min}^{C_N}(\text{FIFO}, \text{FWF}) \leq -1 + \tfrac{r}{k}$$

Proof We define $J_n = \langle P, B^n \rangle$, where $P = \langle 1, 2, \ldots, k, \ldots N, 1, 2, \ldots, r-1 \rangle$ and B is defined by

$$B = \left[\begin{array}{cccc|ccccc} r & r-1 & \cdots & 1 & N & N-1 & \cdots & 2r+2 & 2r+1 \\ 2r & 2r-1 & \cdots & r+1 & r & r-1 & \cdots & 3r+2 & 3r+1 \\ 3r & 3r-1 & \cdots & 2r+1 & 2r & 2r-1 & \cdots & 4r+2 & 4r+1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ N & N-1 & \cdots & k+1 & k & k-1 & \cdots & r+2 & r+1 \end{array}\right]$$

The vertical line is merely for reference in the proof.

Let R denote the number of rows in B. Note that R is $\frac{\mathrm{LCM}(N,r)}{r}$, where $\mathrm{LCM}(N,r)$ denotes the least common multiple of N and r. There are r columns before and $k-r$ columns after the vertical line. Thus, $|J_n| = N + r - 1 + kRn$.

Observe that the sequence "turns" exactly once, namely at the first request after P. There are k hits following P for LRU. After that, the sequence moves around the cycle, so LRU faults on all of these requests, giving a total cost of $\mathrm{LRU}(J_n) = N + r - 1 + kRn - k$. Note that FWF faults on the same requests as LRU, except that it could have fewer hits immediately following P, so $\mathrm{FWF}(J_n) \geq \mathrm{LRU}(J_n)$.

For FIFO, when processing $\langle k+1, \ldots, N\rangle$ in P, it evicts the pages $\{1, \ldots, r\}$, and then when processing $\langle 1, 2, \ldots, r-1\rangle$, it evicts the pages $\{r+1, \ldots, 2r-1\}$. Then, at the very first request of B, it incurs the next fault and evicts page $2r$. After that, the set of pages outside its cache is $\{r+1, \ldots, 2r\}$, and FIFO does not fault again in the first row of B. FIFO then faults on the first r requests in the second row, evicting $\{2r+1, \ldots, 3r\}$. This pattern continues, so FIFO only faults on the first r entries in each row of B. Therefore, $\mathrm{FIFO}(J_n) = N + rRn$.

For $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FWF}\}$, this gives

$$\begin{aligned}
\mathrm{Min}^{C_N}(\mathrm{FIFO}, \mathcal{A}) &\leq \lim_{n\to\infty} \frac{\mathrm{FIFO}(J_n) - \mathrm{LRU}(J_n)}{|J_n|} \\
&= \lim_{n\to\infty} \frac{N + rRn - (N + r - 1 + kRn - k)}{N + r - 1 + kRn} \\
&= -\frac{k-r}{k} = -1 + \frac{r}{k}.
\end{aligned}$$

□

Theorem 8. For the cycle access graph C_N, where $N = k + r$, $1 \leq r < k$,

$$\left[-1 + \frac{r}{k}, \frac{1}{2} - \frac{1}{4k-2}\right] \subseteq \mathcal{I}^{C_N}[\mathrm{FIFO}, \mathrm{LRU}] \subseteq \left[-1 + \frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}\right]$$

Proof The left-most containment follows from Lemmas 14 and 13, and the right-most from Lemma 5. □

We define $r = N - k$ and use this definition in the following results involving FAR on cycles. For $r \geq k$, we have a cycle of length at least $2k$, so the cache page that is farthest from the current fault is at least k pages away, in either direction. This is why LRU and FAR^{C_N} evict the exact same page at every fault, making them identical when $N \geq 2k$, so only the range $1 \leq r < k$ is interesting. Thus, we assume that $1 \leq r < k$ in the following.

The exact results to be presented sometimes depend on the relationship between k and N, e.g., whether or not r divides N (denoted $r \mid N$). To express many of the results, we need the following term that, for brevity, we will simply denote X_r:

$$X_r = r(x-1) + \left\lceil \frac{N}{2^x} \right\rceil, \text{ where } x = \left\lfloor \log \frac{N}{r} \right\rfloor$$

In the following lemma, we analyze the behavior of FAR^{C_N} on the simplest sequence exploiting the cycle structure.

Lemma 15. For FAR^{C_N} and the sequence $I_n = \langle 1, 2, \ldots, k, \ldots, N \rangle^n$ in C_N, each k-phase, except the first and possibly the last, has X_r faults, and

$$\left\lfloor \frac{nN}{k} \right\rfloor X_r + k - X_r \leq \text{FAR}^{C_N}(I_n) \leq \left\lfloor \frac{nN}{k} \right\rfloor X_r + k - 1.$$

Proof In the given sequence, as in any other sequence, the first k-phase contributes k faults. The first phase change in I_n occurs on page $k+1$, at which all the other $N-1$ pages are unmarked. Given that the sequence goes around the cycle n times, without turning, the properties discussed about faults in the second phase holds for all subsequent ones, with the possible exception of the last which may contain just one fault. Consider the fault incurred at the phase change at page $k+1$. The page evicted lies in the middle of the unmarked segment of pages $[k+2, \ldots, N, 1, \ldots, k]$. Following this, there are $r-1$ more faults before the next hit. Each fault leads to the eviction of the page adjacent to the most recently evicted page, the evictions moving in the same direction in which the faults are encountered.

In each phase, we refer to the first r faults as the first *batch*, faults numbered $r+1$ through $2r$ as the second batch, and so on. If there are i batches of faults in one k-phase, then the first $i-1$ batches will contribute r faults each, and the last batch will have at least one and at most r faults. For the ith batch, we denote the length of the unmarked segment after marking the first page in the batch by d_i, and the distance to the page evicted at the first fault in the ith batch by D_i. These distances are measured in the direction in which the faulting page was approached. Therefore, $d_1 = N - 1$ and for $i \geq 1$, $d_{i+1} = d_i - D_i$. Since LRU is used to break ties, if for some i, d_i is even, then the closer of the two midpoints is evicted at the first fault of the ith batch. Thus, we have the following dependencies:

$$\text{For } i \geq 1,\ D_i = \lceil d_i/2 \rceil \text{ and } d_{i+1} = d_i - D_i = \lfloor d_i/2 \rfloor$$

From the recurrence $d_i = \lfloor d_{i-1}/2 \rfloor$, we obtain the following relation:

$$\text{For } i \geq 1,\ d_i = \left\lfloor \frac{d_{i-1}}{2} \right\rfloor = \left\lfloor \frac{1}{2} \left\lfloor \frac{d_{i-2}}{2} \right\rfloor \right\rfloor = \left\lfloor \frac{d_{i-2}}{2^2} \right\rfloor = \left\lfloor \frac{d_1}{2^{i-1}} \right\rfloor = \left\lfloor \frac{N-1}{2^{i-1}} \right\rfloor$$

A k-phase ends when all the pages in the cache are marked and the next request will be a fault. At any given instant, the marked segment is a path in C_N. This implies that a phase ends when the r pages outside the cache constitute the unmarked segment, and one of those unmarked pages is requested. Therefore, if there are i batches in a k-phase, then $d_i + 1 \leq 2r$. Stated differently, the smallest value of i for which $d_i + 1 \leq 2r$ gives the number of batches in a phase.

If there is an i such that $d_i + 1 = 2r$, then the phase has i batches contributing r faults each. Otherwise, if $d_i + 1 < 2r$, then the first $i - 1$ batches contribute r faults each and the last batch contributes fewer than r.

It follows from the above that $d_i + 1 = \lceil \frac{N}{2^{i-1}} \rceil$. Solving $\lceil \frac{N}{2^{i-1}} \rceil \leq 2r$ gives $i - 1 = \lfloor \log \frac{N}{r} \rfloor$ batches with r faults each and the last with $y = \lceil \frac{N}{2^{i-1}} \rceil - r$ faults. Therefore, each phase in I_n, excluding the first and perhaps the last, contains $r(i-1) + y$ faults. There are $\lfloor \frac{nN}{k} \rfloor$ complete phases in sequence I_n and if the last phase is not complete, that is, $k \nmid nN$, then the last phase can contain at most $r(i-1) + y - 1$ faults. Thus, we obtain the following relation for FAR^{C_N} serving I_n:

$$\left\lfloor \frac{nN}{k} \right\rfloor (rx + y) + c \leq \text{FAR}^{C_N}(I_n) \leq \left\lfloor \frac{nN}{k} \right\rfloor (rx + y) + rx + y - 1 + c,$$

where $x = \lfloor \log \frac{N}{r} \rfloor$, $y = \lceil \frac{N}{2^x} \rceil - r$ and $c = k - (rx + y)$. □

The following lemma analyzes the behavior of FAR^{C_N} when the cycle structure is *not* used in the request sequence. Thus, the cycle access graph is used as a path access graph. However, FAR^{C_N} is oblivious to this and uses distances involving the non-utilized edge in the graph, leading to non-optimal results.

From now on, whenever needed , we use $\hat{N}$ to denote N, if N is even, and $N-1$, otherwise.

Lemma 16. For FAR^{C_N} and the sequence $I_n = \langle 1, 2, \ldots, k, \ldots, N-1, N, N-1, \ldots, 2\rangle^n$ in C_N, each k-phase, except the first (which has k) and the last (which has r), has $rx + y$ faults, where $x = \left\lfloor \log \frac{\hat{N}}{r} \right\rfloor$ and $y = \left\lfloor \frac{\hat{N}}{2^x} \right\rfloor - r$.

Proof The first k-phase in sequence I_n has k faults. In any k-phase of I_n, excluding the first, the first set of r faults is called the first *batch*, faults numbered $r+1$ through $2r$ is called the second batch, and so on. If there are i batches of faults in one k-phase, then the first $i-1$ batches will contribute r faults each, and the last batch will have at least one and at most r faults.

As before, the length of the unmarked segment after marking the first page of the ith batch is denoted by d_i and the page located D_i pages away is evicted at that fault. All these distances are measured in the direction in which the first fault of the batch was encountered. Note that within each iteration within I_n, there are two phase changes, occurring first at $k+1$ and then at r. In the following discussion, we explain the behavior of FAR^{C_N} in one iteration within I_n. Since the same properties hold for others, that will lead to a bound for $\text{FAR}^{C_N}(I_n)$.

At the end of a phase and right before the start of the next, FAR^{C_N}'s cache is connected. Hence, the r pages outside the cache also form a connected component, implying that the sets of pages outside FAR^{C_N}'s cache immediately before the phase changes at page $k+1$ and page r are $\{k+1, \ldots, N\}$ and $\{r, r-1, \ldots, 1\}$, respectively.

For the phase changes at page $k+1$ and page r, the faulting request is approached from page k and page $r+1$ in the access graph, respectively. For either case, we have $d_1 = N-1$ and as in Lemma 15, the page located $D_1 =$

$\lceil d_1/2 \rceil$ vertices away is evicted at the first fault in the phase. The next $r-1$ faults lead to eviction of pages in the same direction in which the faults are encountered. Unlike in the previous lemma, the sequence considered here turns back at the end of the first batch and so the second batch of faults start at the most recently evicted page.

Phase change at page $k+1$: The first fault in the second batch occurs when the sequence reaches D_1, which is also the first page marked in the batch. The unmarked segment at that instant is $\{D_1 - 1, D_1 - 2, \ldots, 1\}$.

Phase change at page r: Analogously to the previous case, the second batch of faults starts when the sequence reaches $N - D_1 + 1$. The unmarked segment at that instant is

$$\Big[N - D_1 + 2, N - D_1 + 3, \ldots, N - D_1 + r, \ldots, k, k+1, \ldots, N-1, N\Big].$$

In either case, the length of the unmarked segment is $d_2 = D_1 - 1$. Note that for both locations of phase change, the change in direction of the sequence right after the first batch affects the resolution of ties in subsequent batches. In fact, if d_2 is even, then the farther of the two midpoints, measured in the same direction as the fault, is less recently requested than the other. Therefore, for each phase, we have the following correspondence:

$$D_2 = \begin{cases} d_2/2 + 1, & \text{if } d_2 \text{ is even} \\ \lceil d_2/2 \rceil, & \text{if } d_2 \text{ is odd} \end{cases}$$

Since, in either case, from the second batch onwards, the sequence does not change direction for the rest of the phase, all subsequent ties within the phase are resolved in the manner of the second batch. Therefore, in any given phase, from the second batch onwards, if the unmarked segment is even, the farther of the two midpoints, measured in the same direction in which the fault was approached is evicted in favor of the other. This yields the following set of relations: $d_1 = N - 1$, $D_1 = \lceil d_1/2 \rceil$, $d_2 = D_1 - 1$, and for $i \geq 2$,

$$D_i = \begin{cases} d_i/2 + 1, & \text{if } d_i \text{ is even} \\ \lceil d_i/2 \rceil, & \text{if } d_i \text{ is odd} \end{cases}$$

and

$$d_{i+1} = d_i - D_i = \begin{cases} d_i/2 - 1, & \text{if } d_i \text{ is even} \\ \lfloor d_i/2 \rfloor, & \text{if } d_i \text{ is odd} \end{cases}$$

This implies that for all $i \geq 2$, $d_{i+1} = \left\lfloor \frac{d_i - 1}{2} \right\rfloor$.

We now establish the following claim. Recall that $\hat{N}$ denotes N, if N is even, and $N-1$, otherwise.

Claim: For $i \geq 3$, we have $d_i + 1 = \left\lfloor \frac{D_1}{2^{i-2}} \right\rfloor = \left\lfloor \frac{\hat{N}}{2^{i-1}} \right\rfloor$.

Since $D_1 = \left\lceil \frac{N-1}{2} \right\rceil$, using the new notation, $D_1 = \frac{\hat{N}}{2}$.

We proceed to show by induction that for $i \geq 3$, $d_i + 1 = \left\lfloor \frac{D_1}{2^{i-2}} \right\rfloor$.

For the base case, $i = 3$, we have

$$d_3 = \left\lfloor \frac{d_2 - 1}{2} \right\rfloor = \left\lfloor \frac{(D_1 - 1) - 1}{2} \right\rfloor = \left\lfloor \frac{D_1}{2} \right\rfloor - 1.$$

Hence, $d_3 + 1 = \left\lfloor \frac{D_1}{2^{3-2}} \right\rfloor$.

Now, we assume that the induction hypothesis holds up to some $i \geq 3$. For the induction step, we prove the relation $d_{t+1} = \left\lfloor \frac{d_t - 1}{2} \right\rfloor$, by applying the hypothesis for d_t in the last equality below.

$$d_{t+1} = \left\lfloor \frac{d_t - 1}{2} \right\rfloor = \left\lfloor \frac{1}{2}(d_t + 1) - 1 \right\rfloor = \left\lfloor \frac{1}{2} \left\lfloor \frac{D_1}{2^{t-2}} \right\rfloor \right\rfloor - 1$$

Therefore, $d_{t+1} + 1 = \left\lfloor \frac{D_1}{2^{t-1}} \right\rfloor$, and the claim is proved.

As was the case in the previous lemma, the last batch starts when, for the first time in the current phase, the length of the unmarked segment is no greater than $2r$, i.e., the smallest value i for which $d_i + 1 \leq 2r$ gives the number of batches in the phase. Solving $\left\lfloor \frac{\hat{N}}{2^{i-1}} \right\rfloor \leq 2r$ gives $i - 1 = \left\lfloor \log \frac{\hat{N}}{r} \right\rfloor$. Therefore, the first $i-1$ batches in a k-phase have r faults each. In the last batch, though, there are exactly $\left\lfloor \frac{\hat{N}}{2^{i-1}} \right\rfloor - r$ faults.

Right before the start of the ith batch, the length of the unmarked segment is d_i+1. The phase must end when the length of the unmarked segment becomes r. Therefore, $d_i + 1 - r$ is an upper bound on the number of faults incurred in the ith batch. □

Note that in the above proof, making the sequence go only up to some other value between $k+1$ and $N-1$, instead of up to N, would never give more faults.

Lemma 17. For $1 \leq r \leq k-1$, in any sequence respecting the cycle access graph C_N, the maximum number of faults incurred by FAR^{C_N} in a k-phase, excluding the first, is at most X_r. In particular, FAR^{C_N} incurs the maximum number of faults in a k-phase if the sequence takes the shortest path between any two faults in that phase. Consequently, in C_{k+1}, each k-phase can generate at most $\lceil \log(k+1) \rceil$ faults for FAR^{C_N}.

Proof Given the eviction rule of FAR^{C_N} in C_N, which is that it evicts the midpoint of the current unmarked segment, it follows that when a sequence does not turn inside a phase, it is taking the shortest path to the next fault. This situation is analyzed in Lemma 15. When a sequence turns such that at least one page is marked before the next turn, then all those pages become unavailable for eviction for the remainder of the phase. A phase ends when all the pages in the cache are marked and a new phase starts at the next fault. Therefore, if a sequence keeps moving along the shortest path which takes it to the next fault, then it is also marking the fewest number of pages in order to get to the next fault, thereby, maximizing the number of faults FAR^{C_N} incurs in the current phase. Hence, the maximum number of faults incurred by FAR^{C_N} in each phase, excluding the first, is upper bounded by X_r, as proved in Lemma 15. The special case of C_{k+1} is given by $r=1$ and so the lemma is proved. □

Lemma 18. For the cycle access graph C_N, and $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FIFO}, \mathrm{FWF}\}$,

$$\mathrm{Min}^{C_N}(\mathcal{A}, \mathrm{FAR}^{C_N}) \geq -\frac{X_r - 1}{k}.$$

Proof Consider an arbitrary sequence I_n in C_N, where n denotes the number of k-phases in the sequence. The last phase of a sequence may contain fewer than k distinct pages, and in that case we can ignore the last phase in I_n. Note that each phase contains requests to k distinct pages. It follows that each phase in a sequence is of length at least k. By Lemma 17, we know that FAR^{C_N} can incur at most X_r faults for each phase, excluding the first. By Lemma 1, $\mathcal{A}$ faults at least once in each phase. In the first phase both algorithms incur k faults. Thus, in each phase, the absolute value of the maximum difference in faults is at most $X_r - 1$. Thus, $\lim_{n\to\infty} \frac{\mathcal{A}(I_n) - \mathrm{FAR}^{C_N}(I_n)}{|I_n|} \geq -\frac{X_r - 1}{k}$. $\square$

Lemma 19. For the cycle access graph C_N,

$$\mathrm{Min}^{C_N}(\mathrm{FIFO}, \mathrm{FAR}^{C_N}) \leq -\frac{X_r - r}{k}.$$

Proof Recall the sequence J_n from the proof of Lemma 14.

$$J_n = \langle P, B^n \rangle, \text{ where } P = \langle 1, 2, \ldots, k, k+1, \ldots, N, 1, 2, \ldots, r-1 \rangle$$

and

$$B = \left[\begin{array}{cccc|ccccc} r & r-1 & \cdots & 1 & N & N-1 & \cdots & 2r+2 & 2r+1 \\ 2r & 2r-1 & \cdots & r+1 & r & r-1 & \cdots & 3r+2 & 3r+1 \\ 3r & 3r-1 & \cdots & 2r+1 & 2r & 2r-1 & \cdots & 4r+2 & 4r+1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ N & N-1 & \cdots & k+1 & k & \vdots & \cdots & r+2 & r+1 \end{array} \right]$$

$|J_n| = kRn + N + r - 1$ and $\mathrm{FIFO}(J_n) = N + rRn$.

There is exactly one turn in J_n, which occurs at the first request after P. For the rest of the sequence, it moves around the cycle without turning. Hence, the number of faults incurred by FAR^{C_N} in each phase of B, excluding the first two, is given by Lemma 15, to be $X_r = r(x-1) + \lceil \frac{N}{2^x} \rceil$, where $x = \lfloor \log \frac{N}{r} \rfloor$. Therefore, $\mathrm{FAR}^{C_N}(J_n) = \lfloor \frac{nkR}{k} \rfloor X_r + c$, where c is a constant. The constant bounds the number of faults in the first two and last phases. Now, $\mathrm{Min}^{C_N}(\mathrm{FIFO}, \mathrm{FAR}^{C_N})$ is at most

$$\lim_{n\to\infty} \frac{\mathrm{FIFO}(J_n) - \mathrm{FAR}^{C_N}(J_n)}{|J_n|} = \lim_{n\to\infty} \frac{nR(r - X_r)}{nRk + N + r - 1} = -\frac{X_r - r}{k}$$

$\square$

Lemma 20. For the cycle access graph C_N, and $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FIFO}, \mathrm{FWF}\}$,

$$\mathrm{Max}^{C_N}(\mathcal{A}, \mathrm{FAR}^{C_N}) \geq 1 - \frac{X_r}{k}.$$

Proof Consider the sequences $I_n = \langle 1, 2, \ldots, N\rangle^n$ in C_N such that k divides nN. It is easy to see that $\mathcal{A}(I_n) = |I_n| = Nn$. By Lemma 15, we have $\mathrm{FAR}^{C_N}(I_n) \leq \frac{Nn}{k}X_r + k - 1$. Thus, $\mathrm{Max}^{C_N}(\mathcal{A}, \mathrm{FAR}^{C_N})$ is at least

$$\lim_{n\to\infty} \frac{\mathcal{A}(I_n) - \mathrm{FAR}^{C_N}(I_n)}{|I_n|} \geq \lim_{n\to\infty} \frac{nN - \frac{nN}{k}X_r - (k-1)}{nN} = 1 - \frac{X_r}{k}$$

□

Lemma 21. For the cycle access graph C_N,

$$\mathrm{Min}^{C_N}(\mathrm{LRU}, \mathrm{FAR}^{C_N}) \leq -\frac{r\left(\left\lfloor \log \frac{\hat{N}}{r} \right\rfloor - 1\right)}{N-1}$$

where $\hat{N}$ is N and $N-1$ if N is even and odd, respectively.

Proof Consider the sequence $I_n = \langle 1, 2, \ldots, N-1, N, N-1, \ldots, 2\rangle^n$ used in the proof of Lemma 16. Clearly, $\mathrm{LRU}(I_n) = 2nr + k - 1$ and $|I_n| = 2(N-1)n$. There are two phase changes in each iteration of I_n, so by Lemma 16,

$$k + 2n\left(r\left\lfloor \log \frac{\hat{N}}{r} \right\rfloor + \left\lfloor \frac{\hat{N}}{2^x} \right\rfloor - r\right) \leq \mathrm{FAR}^{C_N}(I_n),$$

where $x = \left\lfloor \log \frac{\hat{N}}{r} \right\rfloor$.

Now, since $r \leq \left\lfloor \frac{\hat{N}}{2^x} \right\rfloor$,

$$\lim_{n\to\infty} \frac{\mathrm{LRU}(I_n) - \mathrm{FAR}^{C_N}(I_n)}{|I_n|} \leq -\frac{r\left(\left\lfloor \log \frac{\hat{N}}{r} \right\rfloor - 1\right)}{N-1}$$

□

When $r = 1$, we get the bound $\mathrm{Min}^{C_{k+1}}(\mathrm{LRU}, \mathrm{FAR}^{C_{k+1}}) \leq -\frac{\lfloor \log k \rfloor - 1}{k}$.

Theorem 9. For the cycle access graph C_N,

$$\left[-\frac{X_r - r}{k}, 1 - \frac{X_r}{k}\right] \subseteq \mathcal{I}^{C_N}[\mathrm{FIFO}, \mathrm{FAR}^{C_N}] \subseteq \left[-\frac{X_r - 1}{k}, 1 - \frac{1}{k}\right],$$

$$\left[-\frac{r\left(\left\lfloor \log \frac{\hat{N}}{r} \right\rfloor - 1\right)}{N-1}, 1 - \frac{X_r}{k}\right] \subseteq \mathcal{I}^{C_N}[\mathrm{LRU}, \mathrm{FAR}^{C_N}] \subseteq \left[-\frac{X_r - 1}{k}, 1 - \frac{1}{k}\right]$$

and

$$\left[0, 1 - \frac{X_r}{k}\right] \subseteq \mathcal{I}^{C_N}[\mathrm{FWF}, \mathrm{FAR}^{C_N}] \subseteq \left[0, 1 - \frac{1}{k}\right].$$

Proof The first relation follows from Lemma 4 and Lemmas 18, 19, and 20, and the second from Lemma 4 and Lemmas 18, 21, and 20. The third result follows from Lemma 4 and Lemmas 6 and 20. □

Having established all the results, we list a technical overview in Table 1, in the form of a condensation of our results. In our theorems, we have given bounds as exact as possible. In the table, however, we have used asymptotic notation in a few places where we believe that it improves readability. This has enabled us to present results for the general case, paths, and stars in a very readable manner, whereas the inherent nature of the problem for cycles makes it difficult to improve readability without sacrificing too much with regards to precision.

7. Concluding Remarks

Relative interval analysis has the advantage that it can separate algorithms properly when one algorithm is at least as good as another on every sequence and is better on some. This was reflected in the results concerning FWF which is dominated by the other algorithms considered for all access graphs. It was also reflected by the result showing that LRU and FAR^{P_N} have better performance than FIFO on paths, P_N. The analysis also found the expected result that FAR^G, which is designed to perform well on access graphs, performs better than both LRU and FIFO on cycles, i.e., when G is C_N.

However, it is disappointing that the relative interval analysis of LRU and FIFO on stars and cycles found that FIFO had the better performance, confirming the original results by [13] on complete graphs. The use of the absolute value of the difference in fault rate allows the length of the sequences to play a critical role in relative interval analysis. As shown in our results, when LRU outperforms FIFO (in terms of the total number of faults across the whole sequence), the length of the sequence is asymptotically longer than the sequences in which FIFO outperforms LRU. Consequently, the fault rate measure is favorable towards FIFO. The conclusion that relative interval analysis favours

FIFO to LRU is attributable to this.

Nevertheless, we believe that the access graph technique is one of the important techniques for defining input sequences with locality of reference. Possibly, seeing how well a performance measure works in combination with access graphs is one reasonable test of the general applicability of a performance measure. It has been proven to work well in combination with competitive analysis as well as relative worst order analysis, but, as we have seen, less well with relative interval analysis. To try to understand other quality measures for online algorithms better, it would be interesting to determine which measures work well in combination with access graphs by considering the separation results such analyses would give rise to.

Acknowledgments

We would like to thank anonymous referees for improving the presentation of the results of this paper.

References

[1] Susanne Albers, Lene M. Favrholdt, and Oliver Giel. On paging with locality of reference. *Journal of Computer and System Sciences*, 70(2):145–175, 2005.

[2] Spyros Angelopoulos, Reza Dorrigiv, and Alejandro López-Ortiz. On the separation and equivalence of paging strategies. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 229–237, 2007.

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

[4] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.

[5] Joan Boyar and Lene M. Favrholdt. The relative worst order ratio for on-line algorithms. *ACM Transactions on Algorithms*, 3(2), 2007. Article No. 22.

[6] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. The relative worst order ratio applied to paging. *Journal of Computer and System Sciences*, 73(5):818–843, 2007.

[7] Joan Boyar, Sushmita Gupta, and Kim S. Larsen. Access graphs results for LRU versus FIFO under relative worst order analysis. In *Thirteenth Scandinavian Symposium and Workshops on Algorithm Theory*, volume 7357 of *Lecture Notes in Computer Science*, pages 328–339. Springer, 2012.

[8] Joan Boyar, Sandy Irani, and Kim S. Larsen. A comparison of performance measures for online algorithms. In *Eleventh International Algorithms and Data Structures Symposium*, volume 5664 of *Lecture Notes in Computer Science*, pages 119–130. Springer, 2009. To appear in *Algorithmica*.

[9] Marek Chrobak and John Noga. LRU is better than FIFO. *Algorithmica*, 23(2):180–185, 1999.

[10] Peter J. Denning. The working set model for program behaviour. *Communications of the ACM*, 11(5):323–333, 1968.

[11] Peter J. Denning. Working sets past and present. *IEEE Transactions on Software Engineering*, 6(1):64–84, 1980.

[12] Reza Dorrigiv and Alejandro López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.

[13] Reza Dorrigiv, Alejandro López-Ortiz, and J. Ian Munro. On the relative dominance of paging algorithms. *Theoretical Computer Science*, 410:3694–3701, 2009.

[14] Martin R. Ehmsen, Jens S. Kohrt, and Kim S. Larsen. List factoring and relative worst order analysis. *Algorithmica*, 66(2):287–309, 2013.

[15] Amos Fiat and Anna R. Karlin. Randomized and multipointer paging with locality of reference. In *Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 626–634. ACM, 1995.

[16] Amos Fiat and Manor Mendel. Truly online paging with locality of reference. In *38th Annual Symposium on Foundations of Computer Science*, pages 326–335. IEEE Computer Society, 1997.

[17] Amos Fiat and Ziv Rosen. Experimental studies of access graph based heuristics: Beating the LRU standard? In *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 63–72. ACM/SIAM, 1997.

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

[19] Anna R. Karlin, Steven J. Phillips, and Prabhakar Raghavan. Markov paging. *SIAM Journal on Computing*, 30(3):906–922, 2000.

[20] Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):300–317, 2000.

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

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

[23] E. Torng. A unified analysis of paging and caching. *Algorithmica*, 20:175–200, 1998.

[24] Neal E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11:525–541, 1994.

[25] Neal E. Young. On-line paging against adversarially biased random inputs. Technical Report arXiv:cs/0205007 [cs.DS], arXiv, 2002.

Table 1: Summary of Results

Lower Bound	Relative Interval	Upper Bound	
	$\mathcal{I}[\text{FIFO}, \text{LRU}] =$	$\left[-1+\frac{1}{k},\ \frac{1}{2}-\frac{1}{4k-2}\right]$	(T1)
	$\mathcal{I}[\text{FWF}, \mathcal{A}] =$	$\left[0,\ 1-\frac{1}{k}\right]$	(C1)
	$\mathcal{I}[\text{FWF}, \text{FIFO}] =$	$\left[0,\ 1-\frac{1}{k}-O(\frac{1}{k^2})\right]$	(T3)
	$\mathcal{I}^{P_N}[\text{FIFO}, \mathcal{A}] =$	$\left[0,\ \frac{1}{2}-\frac{1}{2k}\right]$	(T4)
	$\mathcal{I}^{P_N}[\text{FWF}, \mathcal{A}] =$	$\left[0,\ 1-\frac{1}{k}\right]$	(T2)
	$\mathcal{I}^{P_N}[\text{FWF}, \text{FIFO}] =$	$\left[0,\ 1-\frac{1}{k}-O(\frac{1}{k^2})\right]$	(T3)
	$\mathcal{I}^{S_N}[\text{FIFO}, \mathcal{A}] =$	$\left[-\frac{1}{2}+\frac{1}{2(k-1)}+\frac{1}{2k(k-1)},\ \frac{1}{4}+\frac{1}{8k-12}\right]$	(T5)
	$\mathcal{I}^{S_N}[\text{FWF}, \mathcal{A}] =$	$\left[0,\ \frac{1}{2}\right]$	(T6)
	$\mathcal{I}^{S_N}[\text{FWF}, \text{FIFO}] =$	$\left[0,\ \frac{1}{2}-\frac{1}{2k(k-1)}\right]$	(T6)
$\left[-1+\frac{r}{k},\ \frac{1}{2}-\frac{1}{4k-2}\right]$	$\subseteq \mathcal{I}^{C_N}[\text{FIFO}, \text{LRU}] \subseteq$	$\left[-1+\frac{1}{k},\ \frac{1}{2}-\frac{1}{4k-2}\right]$	(T8)
$\left[-\frac{X_r-r}{k},\ 1-\frac{X_r}{k}\right]$	$\subseteq \mathcal{I}^{C_N}[\text{FIFO}, \text{FAR}] \subseteq$	$\left[-\frac{X_r-1}{k},\ 1-\frac{1}{k}\right]$	(T9)
$\left[-\frac{r\left(\left\lfloor\log\frac{\hat{N}}{r}\right\rfloor-1\right)}{N-1}, 1-\frac{X_r}{k}\right]$	$\subseteq \mathcal{I}^{C_N}[\text{LRU}, \text{FAR}] \subseteq$	$\left[-\frac{X_r-1}{k},\ 1-\frac{1}{k}\right]$	(T9)
	$\mathcal{I}^{C_N}[\text{FWF}, \text{LRU}] =$	$\left[0,\ 1-\frac{1}{k}\right]$	(T7)
$\left[0,\ 1-\frac{X_r}{k}\right]$	$\subseteq \mathcal{I}^{C_N}[\text{FWF}, \text{FAR}] \subseteq$	$\left[0,\ 1-\frac{1}{k}\right]$	(T9)
	$\mathcal{I}^{C_N}[\text{FWF}, \text{FIFO}] =$	$\left[0,\ 1-\frac{1}{k}-O(\frac{1}{k^2})\right]$	(T3)

The last column refers to the theorem or corollary establishing the result.

$\mathcal{A} \in \{\text{FAR}, \text{LRU}\}$ and $\mathcal{B} \in \{\text{FAR}, \text{FIFO}, \text{LRU}\}$.

FAR is with respect to the access graph given as superscript to $\mathcal{I}$.

For results involving r, $r = N-k$, $1 \leq r \leq k-1$, $X_r = r(x-1) + \left\lceil \frac{N}{2^x} \right\rceil$ with $x = \left\lfloor \log \frac{N}{r} \right\rfloor$.

$\hat{N}$ denotes N if N is even, and $N-1$ otherwise.