Relative Interval Analysis of Paging Algorithms on Access Graphs,*

Joan Boyar Sushmita Gupta Kim S. Larsen
University of Southern Denmark
Odense, Denmark
{joan,sgupta,kslarsen}@imada.sdu.dk

Abstract

Access graphs, which have been used previously in connection with 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 expected results are obtained. 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.

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. When a page is requested, it may already be in cache (called a "hit"), or it must be brought into cache (called a "fault"). The algorithmic problem is the one of choosing an eviction strategy, i.e., which page to evict from cache in the case of a fault, with the objective of minimizing the total number of faults.

*A preliminary version of this paper will appear in the proceedings of the Thirteenth Algorithms and Data Structures Symposium. Supported in part by the Danish Council for Independent Research. Part of this work was carried out while the first and third authors were visiting the University of Waterloo.

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, which works with respect to a known access graph. Whenever a page is requested, it is marked. When it is necessary to evict a page, it always evicts an unmarked page. If all pages are marked in such a situation, FAR first unmarks all pages. The unmarked page it chooses to evict is the one farthest from any marked page in the access graph. For breaking possible ties, we assume the LRU strategy in this paper.

Understanding differences in paging algorithms' behavior under various circumstances has been a topic for much research. The most standard measure of quality of an online algorithm, competitive analysis [17, 14], cannot directly distinguish between most of them. It deems LRU, FIFO, and FWF equivalent, with a competitive ratio of k, where k denotes the size of the cache. Other measures, such as relative worst order analysis [5, 6], can be used to obtain more separations, including that LRU and FIFO are better than FWF and that look-ahead helps. No techniques have been able to separate LRU and FIFO, without adding some modelling of locality of reference.

Although LRU performs better than FIFO in some practical situations [18], 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], which basically follows from LRU and FIFO being demand paging algorithms. Thus, it is not surprising that some assumptions involving locality of reference are necessary to separate them.

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 [16] and Markov chain based models [15].

The earlier successes and the generality of access graphs, together with the possibilities the model offers with regards to investigating specific access patterns, makes it an interesting object for further studies. In the light of the recent focus on development of new performance measures, together with the comparative studies initiated in [8], exploring access graphs results in the context of new performance measures seems like a promising direction for expanding our understanding of performance measures as well as concrete algorithms.

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 reflect 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 may or may not be interesting. There are many sequences were FIFO is better than LRU; they just seem to occur less often in real applications.

Comparing two algorithms under almost any analysis technique is generally equivalent to considering them with the complete graph as an access graph, since the complete graph does not restrict the request sequence in any way. Thus, LRU and FIFO are equivalent on complete graphs under both competitive analysis and relative worst order analysis, since they are equivalent without considering access graphs.

In this paper, we consider relative interval analysis [13]. 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. However, it is similar to competitive analysis in that the two algorithms are always compared on exactly the same sequence. 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. 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: $[-1+\frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}] \subseteq \mathcal{I}(\mathrm{FIFO}, \mathrm{LRU})$. We start by proving that this is tight: $\mathcal{I}(\mathrm{FIFO}, \mathrm{LRU}) = [-1+\frac{1}{k}, \frac{1}{2} - \frac{1}{4k-2}]$. 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. Considering complete graphs (or cliques) implies that there are no restrictions on the input sequences, so this is equivalent to considering the situation without an access graph. Table 1 shows our results.

Comparing these 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 best on paths, which is not surprising since it is in fact optimal on paths and FIFO is not. On the other access graphs considered here, relative interval analysis gives results which can be interpreted as incomparability, but leaning towards deeming FIFO the better algorithm. Relative worst order analysis, on the other hand, shows that on cycles, LRU is strictly better than FIFO, and on complete graphs, they are equivalent. It has not yet been studied on stars, but an incomparability result for LRU and FIFO has been found for a family of graphs growing with the length of the input.

Table 1: Summary of Results

Lower Bound	Relative Interval	Upper Bound	Th.
	$\mathcal{I}[\text{FIFO}, \text{LRU}] \quad =$	$\left[-1+\frac{1}{k},\ \frac{1}{2}-\frac{1}{4k-2}\right]$	1
	$\mathcal{I}[\text{FWF}, \mathcal{A}] \quad =$	$\left[0,\ 1-\frac{1}{k}\right]$	2
$\left[0,\ 1-\frac{k+1}{k^2}\right]$	$\subseteq \quad \mathcal{I}[\text{FWF}, \text{FIFO}] \quad \subseteq$	$\left[0,\ 1-\frac{1}{k}\right]$	3
	$\mathcal{I}^{P_N}[\text{FIFO}, \mathcal{A}] \quad =$	$\left[0,\ \frac{1}{2}-\frac{1}{2k}\right]$	4
	$\mathcal{I}^{P_N}[\text{FWF}, \mathcal{A}] \quad =$	$\left[0,\ 1-\frac{1}{k}\right]$	2
$\left[0,\ 1-\frac{k+1}{k^2}\right]$	$\subseteq \mathcal{I}^{P_N}[\text{FWF}, \text{FIFO}] \subseteq$	$\left[0,\ 1-\frac{1}{k}\right]$	3
$\left[-\frac{1}{2}+\Theta(\frac{1}{k}),\ \frac{1}{4}+\Theta(\frac{1}{k})\right]$	$\subseteq \quad \mathcal{I}^{S_N}[\text{FIFO}, \mathcal{A}] \quad \subseteq$	$\left[-\frac{1}{2}+\Theta(\frac{1}{k}),\ \frac{1}{4}+\Theta(\frac{1}{k})\right]$	5
	$\mathcal{I}^{S_N}[\text{FWF}, \mathcal{B}] \quad =$	$\left[0,\ \frac{1}{2}\right]$	6
$\left[-1+\frac{r}{k},\ \frac{1}{2}-\frac{1}{4k-2}\right]$	$\subseteq \mathcal{I}^{C_N}[\text{FIFO}, \text{LRU}] \subseteq$	$\left[-1+\frac{1}{k},\ \frac{1}{2}-\frac{1}{4k-2}\right]$	7
	$\mathcal{I}^{C_N}[\text{FWF}, \text{LRU}] \quad =$	$\left[0,\ 1-\frac{1}{k}\right]$	8
$\left[-\frac{r\left(\left\lfloor \log \frac{\hat{N}}{r} \right\rfloor - 1\right)}{N-1}, 1-\frac{X_r}{k}\right]$	$\subseteq \quad \mathcal{I}^{C_N}[\text{LRU}, \text{FAR}] \quad \subseteq$	$\left[-\frac{X_r-1}{k},\ 1-\frac{1}{k}\right]$	9
$\left[0,\ 1-\frac{X_r}{k}\right]$	$\subseteq \quad \mathcal{I}^{C_N}[\text{FWF}, \text{FAR}] \quad \subseteq$	$\left[0,\ 1-\frac{1}{k}\right]$	9
$\left[0,\ 1-\frac{k+1}{k^2}\right]$	$\subseteq \mathcal{I}^{C_N}[\text{FWF}, \text{FIFO}] \subseteq$	$\left[0,\ 1-\frac{1}{k}\right]$	3

$\mathcal{A} \in \{\text{FAR}, \text{LRU}\}$ and $\mathcal{B} \in \{\text{FAR}, \text{FIFO}, \text{LRU}\}$.
$N = k + r$, with $1 \leq r \leq k-1$, $X_r = r(x-1) + \left\lceil \frac{N}{2^x} \right\rceil$ with $x = \left\lfloor \log \frac{N}{r} \right\rfloor$.
$\hat{N}$ denotes N if N is even, and $N-1$ otherwise.

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. This is the same as the number of different pages we consider. We will always assume that $N > k$, since otherwise the problem is trivial, and let $r = N - k$. A requests 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. □

Thus, 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, focusing on when 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. □

There are some well-known and important classifications of paging algorithms, which are used here and in most other papers on paging [3]: An paging algorithm is called *conservative* 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* 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, and FWF are marking algorithms.

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:

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

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

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

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

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

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

$\mathcal{B}$ *has better performance than* $\mathcal{A}$ if $\mathrm{Max}^G(\mathcal{A},\mathcal{B}) > |\mathrm{Min}^G(\mathcal{A},\mathcal{B})|$.

$\mathcal{B}$ *dominates* $\mathcal{A}$ 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.

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.

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

Lemma 1 Any algorithm has at least $b+k-1$ faults on a sequence consisting of b complete k-phases or b complete k-blocks defined with respect to any conservative or marking algorithm.

Proof Let p be the page requested first in Phase 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 must fault at least once in I'. Thus, an algorithm 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 request in two subsequent k-phases are different. Any conservative or marking algorithm gives rise to such k-blocks. $\square$

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

- $\lim_{n\to\infty} \mathrm{Max}_{\mathcal{A},\mathcal{B}}(n, G) = \infty$,
- 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 the number of complete k-phases or k-blocks in I, and the k-blocks are 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 $j \geq 1$, let I_j be a sequence of length j such that I_j maximizes $\mathcal{A}(I) - \mathcal{B}(I)$ over all sequences of length j.

By construction, $\mathrm{Max}_{\mathcal{A},\mathcal{B}}(n, G) = \max_{|I|=j, I\in\mathcal{L}(G)}\{\mathcal{A}(I) - \mathcal{B}(I)\} = \mathcal{A}(I_j) - \mathcal{B}(I_j) \leq f(b_{I_j})$, and by assumption, $|I_j| \geq g(b_{I_j})$. Thus $\frac{\mathrm{Max}_{\mathcal{A},\mathcal{B}}(n,G)}{|I_j|} \leq \frac{f(b_{I_j})}{g(b_{I_j})}$. Now,

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

The second to last equality holds since $\{b_{I_j} \mid j \geq 1\}$ contains infinitely many values. Assume to the contrary that it had a maximum value $b_{I_{j'}}$ for some j'. That would mean that for any j, $\mathrm{Max}_{\mathcal{A},\mathcal{B}}(|I_j|, G) \leq \max\{f(b) \mid 1 \leq b \leq b_{I_{j'}}\}$, contradicting the assumption of the left-hand expression being unbounded.

The last equality holds since we have assumed that the limit exists. $\square$

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

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

- $\lim_{n\to\infty} \mathrm{Min}_{\mathcal{A},\mathcal{B}}(n, G) = -\infty$,

- 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 the number of complete k-phases or k-block in I, and the k-blocks are 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

As remarked earlier, if the access graph is complete, it incurs no restrictions, so the result of this section is in the same model as [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.

Lemma 4 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 We first consider the Min value. Suppose that a sequence I has b complete k-phases. Since LRU is conservative and a complete k-phase contains k distinct pages, it cannot fault more than $bk+k-1$ times [3]. By Lemma 1, $\mathrm{FIFO}(I) \geq k+b-1$. Thus, $\mathrm{FIFO}(I) - \mathrm{LRU}(I) \geq k+b-1-(bk+k-1) = -b(k-1)$. Each k-phase must have length at least k, and $\lim_{b\to\infty} \frac{-b(k-1)}{bk} = -\frac{k-1}{k}$. Clearly, $\min_{|I|=n, I\in\mathcal{L}(G)}\{\mathrm{FIFO}(I) - \mathrm{LRU}(I)\}$ goes towards $-\infty$ as a function of n (see for instance the family of sequences J_n from Lemma 11). Thus, by Lemma 3, $\mathrm{Min}^G(\mathrm{FIFO},\mathrm{LRU}) \geq -\frac{k-1}{k} = -1+\frac{1}{k}$.

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, $\mathrm{FIFO}(I) \leq bk + (k-1)$. 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 algorithms' behavior on it.

As a first step, with every request on which FIFO faults and LRU has a hit, we associate a distinct request where 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 distinct, 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 page that gets associated with r'' (and r) will be later than r, which is a contradiction.

Thus, if LRU faults α_i times in B_i, by the procedure above, we identify at least $k - \alpha_i$ distinct requests. In total, there are at least $\Sigma_{i=1}^{b}(k - \alpha_i) = bk - \Sigma_{i=1}^{b}\alpha_i$ distinct hits for FIFO in I and, since there are b complete k-blocks, at least bk faults. Thus, the length of I is at least $2bk - \Sigma_{i=1}^{b}\alpha_i$, and

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

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

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

Clearly, $\max_{|I|=n, I \in \mathcal{L}(G)}\{\mathrm{FIFO}(I) - \mathrm{LRU}(I)\}$ is unbounded as a function of n (see for instance the family of sequences I_n in Lemma 9). By Lemma 2, $\mathrm{Max}^G(\mathrm{FIFO}, \mathrm{LRU}) \leq \frac{k-1}{2k-1} = \frac{1}{2} - \frac{1}{4k-2}$, since $\lim_{b\to\infty} \frac{b(k-1)}{b(2k-1)-k+1} = \frac{k-1}{2k-1}$. □

From [13] and Lemma 4, we have the following:

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

The following gives general bounds that are applicable to all pairs of algorithms considered here, though in many cases better bounds are proven later. The proof was essentially given in the first paragraph of the proof of Lemma 4.

Proposition 1 Let $\mathcal{A}$ be a conservative or marking algorithm and $\mathcal{B}$ be any algorithm for paging, then for any access graph G, $\mathrm{Min}^G[\mathcal{B},\mathcal{A}] \geq -1 + \frac{1}{k}$ and $\mathrm{Max}^G[\mathcal{A},\mathcal{B}] \leq 1 - \frac{1}{k}$.

3.1 FWF

FWF performs very badly compared to the other algorithms considered here, LRU, FAR, and FIFO. The following is folklore:

Lemma 5 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, $\mathcal{A}^G(I) \leq \mathrm{FWF}^G(I)$ and so

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

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

The upper bound of $1-\frac{1}{k}$ from Proposition 1 is tight for FWF versus either LRU, for any access graph containing a path on $k+1$ vertices, and it is tight for FWF versus FAR on a path containing at least $k+1$ vertices. Note that a cycle on $k+1$ vertices contains a path on $k+1$ vertices, but FAR does not behave identically on these two graphs.

Theorem 2 For the path access graph P_N, where $N \geq k+1$ (and for LRU for any graph containing P_{k+1}), and $\mathcal{A} \in \{\mathrm{LRU},\mathrm{FAR}\}$,

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

Proof Consider the sequence $I_n = \langle 1,2,\ldots,k,k+1,k,\ldots,2\rangle^n$. For this we have $\mathrm{LRU}(I_n) = \mathrm{FAR}(I_n) = 2n+k-1$, and $\mathrm{FWF}(I_n) = 2kn$. Therefore,

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

By Proposition 1, this gives $\mathrm{Max}^{P_N}(\mathrm{FWF},\mathrm{LRU}) = \mathrm{Max}^{P_N}(\mathrm{FWF},\mathrm{FAR}) = 1 - \frac{1}{k}$. Lemma 5 shows that LRU and FAR dominate FWF. □

The same tight result for FWF versus FIFO almost holds.

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

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

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 then requests $S_1, S_2, ..., S_{h-1}$. This initial part of every sequence in our family of sequences ensures that FIFO's order for faulting 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 fault the same number of times on R. FWF faults $2khn$ times on K_n. On the first request to $k+1$ in I_n, FIFO evicts h. Thus, after the fault on $k+1$, its only fault while going "left" (towards lower page numbers) for the first time in J is on h, and its only fault going "right" is on $h+1$. On the ith iteration ($i \leq h-1$) of J, it faults on $h-i+1$ going left and on $h+i$ going right. On iteration h, it only faults on 1, so FIFO has the same cache configuration immediately after J as it had immediately before. Thus, FIFO has $k+1$ faults on J, giving $(k+1)n$ in all. The number of requests in K_n is $2kn$. Thus, $\lim n \to \infty \frac{\mathrm{FWF}(I_n) - \mathrm{FIFO}(I_n)}{|I_n|} = \frac{2khn-(k+1)n}{2khn} = \frac{k-1}{k}$.

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

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

Lemma 5 shows that FIFO dominates FWF. □

4 Path Graphs

In this section, we analyze path access graphs, P_N, with N vertices. We assume that $N \geq k+1$, since otherwise, results become trivial.

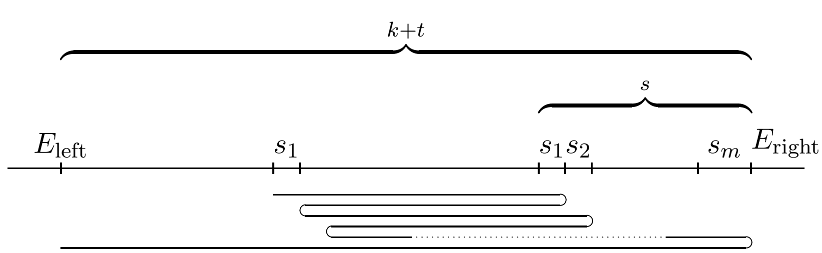
Lemma 6 For the path access graph P_N,

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

Proof Consider any request sequence I. We divide the sequence up into phases as described now (these are *not* k-phases). Initially, define a direction by where LRU makes its kth fault compared with its cache content. Without loss of generality, we assume this happens going to the right on the path.

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 (relatively). 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, though the direction can change, as we will get back to at the end of the proof. The exception in the first phase, adding an extra $k-1$ faults to the cost of LRU as compared with the analysis below, will not influence the result in the 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 $s+t$ 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)st 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 sum up over all phases, equipping each variable with a superscript denoting the phase number.

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 7 For the path access graph P_N,

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

Proof The upper bound was shown in Lemma 6. 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, $\lim_{n\to\infty} \frac{\mathrm{FIFO}(I_n)-\mathrm{LRU}(I_n)}{|I_n|} = \frac{k-1}{2k} = \frac{1}{2} - \frac{1}{2k}$, so the maximum must be at least that large. □

Since LRU is optimal on paths, this gives :

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

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

5 Star Graphs

We let S_N denote a star graph with N vertices. A star graph has a central vertex, s, which is directly connected to $N-1$ other vertices, none of which are directly connected. Thus, we could also see a star graph as a tree with root s and $N-1$ leaves, all located at a distance one from the root. We assume that $N \geq k+1$, since otherwise, results become trivial.

Lemma 8 For the star access graph S_N,

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

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 $2(k-1)$. As in the proof of Lemma 4, FIFO faults at least once for each of these phases. LRU faults only on the leaves and only once on each, so it faults at most $k-1$ times for each phase. Thus, if I has b phases, not counting the first empty phase,, $|I| \geq 2(k-1)(b-1)+1$ and $\mathrm{FIFO}(I) - \mathrm{LRU}(I) \geq (b-1) - (k-1)(b-1) - k = -(k-2)(b-1) - k$, and so $\mathrm{Min}^{S_N}(\mathrm{FIFO}, \mathrm{LRU}) \geq -\frac{k-2}{2(k-1)} = -\frac{1}{2} + \frac{1}{2(k-1)}$.

We will show that the upper bound on $\mathrm{Min}^{S_N}(\mathrm{FIFO}, \mathrm{LRU})$ comes very close to this 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 k does not appear in P and that all the B_i are identical (we use the index for reference). Each $|B_i| = 2k$, so $|I_n| = 2(2k-3) + 2k(k-1)n$. LRU starts B_1 with a fault on the request to k, thereby evicting $k-1$. It then faults on $k-1$ and evicts $k-2$. This repeats and ends with the eviction of k at the request to 1 such that $k-1$ is the least recently used page. Thus, it 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 B_1 as it was at the end of P, the same pattern must be repeated in each B_i. Thus, $\mathrm{LRU}(I_n) = k + (k-1)kn$.

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

Lemma 9 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

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

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

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

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

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 as they only result in hits for both algorithms, while increasing the length of the sequence.

We view 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 s. The number of faults incurred by LRU in 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 s is never evicted from the cache by LRU, we have $1 \leq \alpha_i \leq k-1$.

We now find a lower bound on the length of B_i. First recall that FIFO faults on $k-1$ leaf requests. We now establish some 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. Since LRU incurs α_i faults in B_i, there are at least $k-1-\alpha_i$ distinct leaf requests where LRU has a hit while FIFO faults, ensuring at least $k-1-\alpha_i$ distinct hits for FIFO. Note 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}.

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,

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

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

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

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

$$\begin{aligned}\left[-\tfrac{1}{2}+\tfrac{1}{2(k-1)}, \tfrac{1}{4}+\tfrac{1}{8k-12}\right] &\subseteq \mathcal{I}^{S_N}[\mathrm{FIFO},\mathcal{A}] \\ &\subseteq \left[-\tfrac{1}{2}+\tfrac{1}{2(k-1)}+\tfrac{1}{2k(k-1)}, \tfrac{1}{4}+\tfrac{1}{8k-12}\right]\end{aligned}$$

Proof This follows directly from Lemmas 8 and 9. $\square$

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

Since LRU and FAR perform identically on stars, $\text{Min}^{S_N}(\text{FAR}, \text{LRU}) = \text{Max}^{S_N}(\text{FAR}, \text{LRU}) = 0$.

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

Lemma 10 For the star access graph S_N, and $\mathcal{B} \in \{\text{LRU}, \text{FIFO}\}$,

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

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

Theorem 6 For the star access graph S_N, and $\mathcal{A} \in \{\text{LRU}, \text{FAR}, \text{FIFO}\}$,

$$\mathcal{I}^{S_N}[\text{FWF}, \mathcal{A}] = \left[0, \frac{1}{2}\right].$$

Proof By Lemma 5,

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

Furthermore, since LRU and FAR perform identically on star graphs, we also have that $\text{Min}^{S_N}(\text{FWF}, \text{FAR}) = 0$.

Given any sequence I respecting S_N, it can be viewed a partition of k-phases. Since S_N is a star, each phase must be of length at least $2(k-1)$, and $\mathcal{A}$ must incur at least one fault in each phase. Since FWF can incur at most k faults in each phase, $\lim_{n\to\infty} \frac{\text{FWF}(I)-\mathcal{A}(I)}{|I|} \leq \frac{k-1}{2(k-1)} = \frac{1}{2}$. Hence, $\text{Max}^{S_N}(\text{FWF}, \mathcal{A}) \leq \frac{1}{2}$.

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

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

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

LRU faults on the first request in each B_i. FWF flushes its cache at the start of each B_i. So $|I_n| = 4(k-1)n + 2(k-1), \text{LRU}(I_n) = 2n + k$

and $\text{FWF}(I_n) = 2kn + k$. So $\lim_{n\to\infty} \frac{\text{FWF}(I_n)-\text{LRU}(I_n)}{|I_n|} = \frac{2(k-1)}{4(k-1)} = \frac{1}{2}$ and $\text{Max}^{S_N}(\text{FWF}, \text{LRU}) \geq \frac{1}{2}$.

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

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

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, $\text{FIFO}(B) = k$ and $\text{FIFO}(I_n) = kn + k$. 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 $\text{FWF}(I_n) = k^2n+k$. Therefore, $\lim_{n\to\infty} \frac{\text{FWF}(I_n)-\text{FIFO}(I_n)}{|I_n|} = \frac{k(k-1)}{2(k-1)k} = \frac{1}{2}$ and $\text{Max}^{S_N}(\text{FWF}, \text{FIFO}) \geq \frac{1}{2}$. Since FAR and LRU behave identically on S_N, by Lemma 10, we get $\text{Max}^{S_N}(\text{FWF}, \mathcal{A}) = \frac{1}{2}$. □

6 Cycle Graphs

We consider graphs consisting of exactly one cycle, containing N vertices. We assume that $N \geq k+1$, since otherwise, results become trivial, and define $r = N - k$. We concentrate on the case where $r < k$, since otherwise the cycle is so large that for the algorithms considered here, it works as if it were an infinite path. Thus, for example, there are sequences where FIFO performs worse than LRU, but on worst case sequences, simply going around the cycle, the algorithms perform identically. In 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.

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

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

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

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

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

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 2 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 4. □

We now generalize these results to values of $N = k + r$, where $1 \leq r \leq k-1$.

Lemma 11 For the cycle access graph C_N,

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

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

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

The vertical line is merely for reference in the proof.

Let R denote the number of rows in B. (Note that $R = \frac{LCM(N,r)}{r}$, where $LCM(N, r)$ 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 after the first request in B. There are $k - 1$ hits following that request for both FIFO and LRU. After that, the sequence moves around the cycle, so LRU faults on all of these requests, giving a total cost of $\text{LRU}^{C_N}(J_n) = N + r - k + kRn$. Note that FWF faults on the same requests as LRU, so $\text{FWF}^{C_N}(J_n) = \text{LRU}^{C_N}(J_n)$.

For FIFO, when processing $\langle k+1, \ldots, N\rangle$ in P, it evicts $\{1, \ldots, r\}$, and then when processing $\langle 1, 2, \ldots, r - 1\rangle$, it evicts $\{r + 1, \ldots, 2r - 1\}$. Then, at the very first request of B, it incurs the next fault and evicts $2r$. 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, $\text{FIFO}^{C_N}(J_n) = N + rRn$.

This gives

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

□

Lemma 12 For the cycle access graph C_N,

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

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

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

FIFO faults on the first request in S_1, evicting k, which is requested next. At that request $k-1$ is evicted, leading to a fault on the following request, etc. In total, FIFO faults k times on S_1 and pages were 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,

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

□

Theorem 7 For the cycle access graph C_N,

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

Proof The left-most containment follows from Lemmas 11 and 12, and the right-most from Lemma 4. □

Theorem 8 For the cycle access graph C_N,

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

Proof 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 Proposition 1. The left endpoint is given by Lemma 5. □

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 FAR's behavior on the simplest sequence exploiting the cycle structure.

Lemma 13 For FAR 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

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

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 at $k+1$, 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 $k+1$. The page evicted lies in the middle of the unmarked segment $[k+2, \ldots, N, 1, \ldots, k]$. 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 $d_i + 1 = \left\lceil \frac{N}{2^{i-1}} \right\rceil$. Solving $\left\lceil \frac{N}{2^{i-1}} \right\rceil \leq 2r$ gives $i - 1 = \left\lfloor \log \frac{N}{r} \right\rfloor$ batches with r faults each and the last with $y = \left\lceil \frac{N}{2^{i-1}} \right\rceil - r$ faults. Therefore, each phase in I_n, excluding the first and perhaps the last, contains $r(i-1) + y$ faults. There are $\left\lfloor \frac{nN}{k} \right\rfloor$ complete phases in $I_{r,n}$ 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 serving I_n:

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

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

The following lemma analyzes FAR's behavior on a cycle when the cycle structure is *not* used. Thus, the cycle access graph is used as a path access graph. However, FAR 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 14 For FAR 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 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 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's cache is connected. Hence, the r pages outside the cache also form a connected component, implying that the sets of pages outside FAR's cache immediately before the phase changes at $k+1$ and r are $\{k+1, \dots, N\}$ and $\{r, r-1, \dots, 1\}$, respectively.

For the phase changes at $k+1$ and r, the faulting request is approached from k and $r+1$, respectively. For either case, we have $d_1 = N-1$ and as in Lemma 13, 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 $k+1$: 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 $\{D_1, D_1-1, \dots, 1\}$.

Phase change at 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

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

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

Proof 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 15 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 in a k-phase, excluding the first, is at most X_r. In particular, FAR 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.

Proof Given the eviction rule of FAR 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 13. 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 incurs in the current phase. Hence, the maximum number of faults incurred by FAR in each phase, excluding the first, is upper bounded by X_r, as proved in Lemma 13. The special case of C_{k+1} is given by $r = 1$ and so the lemma is proved. □

Lemma 16 For the cycle access graph C_N, and $\mathcal{A} \in \{\mathrm{LRU}, \mathrm{FIFO}, \mathrm{FWF}\}$,

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

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 and $\mathcal{A}$ incurs at least one fault in each of them. By Lemma 15, we know that FAR can incur at most X_r faults in 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, $\lim_{n\to\infty} \frac{\mathcal{A}(I_n)-\mathrm{FAR}(I_n)}{|I_n|} \geq -\frac{X_r-1}{k}$. □

Lemma 17 For the cycle access graph C_N,

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

Proof Recall the sequence J_n from the proof of Lemma 11.

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

and

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

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

There is exactly one turn in J_n, which occurs at the first request in B and nowhere else. For the rest of the sequence, it moves around the cycle without turning. Hence, the number of faults incurred by FAR in each phase of B, excluding the first two, is given by Lemma 13, to be $X_r = r\big(x-1\big) + \left\lceil \frac{N}{2^x} \right\rceil$, where $x = \left\lfloor \log \frac{N}{r} \right\rfloor$. Therefore, $\mathrm{FAR}^{C_N}(J_n) = \left\lfloor \frac{nkR}{k} \right\rfloor X_r + c$, where c is a constant. The constant bounds the number of faults in the first two phases. Now, $\mathrm{Min}^{C_N}(\mathrm{FIFO}, \mathrm{FAR})$ is at most

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

□

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

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

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 13, we have $\text{FAR}^{C_N}(I_n) \leq \frac{Nn}{k} X_r + k - 1$. Thus, $\text{Max}^{C_N}(\mathcal{A}, \text{FAR})$ is at least

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

□

Lemma 19 For the cycle access graph C_N,

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

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 14. Clearly, $\text{LRU}^{C_N}(I_n) = 2nr + k - 1$ and $|I_n| = 2(N-1)n$. There are two phase changes in each iteration of I_n, so by Lemma 14,

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

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

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

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

□

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

Theorem 9 For the cycle access graph C_N,

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

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

and

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

Proof The first relation follows from Proposition 1 and Lemmas 16, 17, and 18, and the second from Proposition 1 and Lemmas 16, 19, and 18. The third result follows from Proposition 1 and Lemmas 5 and 18. $\square$

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 have better performance than FIFO on paths. The analysis also found the expected result that FAR, which is designed to perform well on access graphs, performs better than both LRU and FIFO on cycles.

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. Clearly, the access graph technique cannot be arbitrarily applied to all quality measures for online algorithms to show that LRU is better than FIFO. To try to understand quality measures better, it would be interesting to determine on which the access graph technique is useful for this well studied problem and on which it is not.

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 (preliminary version). In *23rd Annual ACM Symposium on Theory of Computing*, pages 249–259, 1991.

[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.

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

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

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

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

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