Access Graphs Results for LRU versus FIFO under Relative Worst Order Analysis*

Joan Boyar, Sushmita Gupta, and 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 to model locality of reference in paging, are considered in connection with relative worst order analysis. In this model, FWF is shown to be strictly worse than both LRU and FIFO on any access graph. LRU is shown to be strictly better than FIFO on paths and cycles, but they are incomparable on some families of graphs which grow with the length of the sequences.

1 Introduction

The term *online* algorithm [4] is used for an algorithm that receives its input as a sequence of items, one at a time, and for every item, before knowing the subsequent items, must make an irrevocable decision regarding the current item.

The most standard measure of quality of an online algorithm is *competitive analysis* [17, 22, 20]. This is basically the worst case ratio between the performance of the online algorithm compared to an optimal offline algorithm which is allowed to know the entire input sequence before processing it and is assumed to have unlimited computational power.

Though this measure is very useful and has driven much research, researchers also observed problems [22] with this measure from the beginning: many algorithms obtain the same (poor) ratio, while showing quite different behavior in practice. The *paging problem* is one of the prime examples of these difficulties. The paging problem is the problem of maintaining a subset of a large number of pages in a much smaller, faster cache with space for a limited set of k pages. Whenever a page is requested, it must be brought into cache if it is not already there. In order to make room for such a page, another page currently in cache must be evicted. Therefore, an online algorithm for this problem is often referred to as an eviction strategy.

For a number of years, researchers have worked on refinements or additions to competitive analysis with the aim of obtaining separations between different algorithms for solving an online problem. Some of the most obvious and well-known paging algorithms are the eviction strategies LRU (Least-Recently-Used) and FIFO (First-In/First-Out). One particularly notable result has been the

* Partially supported by the Danish Council for Independent Research.

separation of LRU and FIFO via *access graphs*. Access graphs were introduced in [5] with the aim of modelling the locality of reference that is often seen in real-life paging situations [10, 11]. An access graph is an undirected graph with all pages in slow memory as vertices. Given such a graph, one then restricts the analysis of the performance of an algorithm to sequences respecting the graph, in the sense that any two distinct, consecutive requests must be neighbors in the graph. Important results in understanding why LRU is often observed to perform better than FIFO in practice were obtained in [5, 9], showing that on some access graphs, LRU is strictly better than FIFO, and on no access graph is it worse; all these previous results are with respect to competitive analysis.

Attempts have been made to define new generally-applicable performance measures and to apply measures defined to solve one particular problem more generally to other online problems. A collection of alternative performance measures is surveyed in [12]. Of the alternatives to competitive analysis, *relative worst order analysis* [6] and *extra resource analysis* [19] are the ones that have been successfully applied to most different online problems. See [13] for examples of online problems and references to relative worst order analysis results resolving various issues that are problematic with regards to competitive analysis.

Paging has been investigated under relative worst order analysis in [7]. Separations were found, but LRU and FIFO were proven equivalent, possibly because locality of reference is necessary to separate these two paging algorithms. We apply the access graph technique to relative worst order analysis. Note that the unrestricted analysis in [7] corresponds to considering a complete access graph.

Overall, our contributions are the following. Using relative worst order analysis, we confirm the competitive analysis result [5] that LRU is better than FIFO for path access graphs. Since these two quality measures are so different, this is a a strong indicator of the robustness of the result. Then we analyze cycle access graphs, and show that with regards to relative worst order analysis, LRU is strictly better than FIFO. Note that this does not hold under competitive analysis. The main technical contribution is the proof showing that on cycles, with regards to relative worst order analysis, FIFO is never better than LRU. Clearly, paths and cycles are the two most fundamental building blocks, and future detailed analyses of any other graphs type will likely build on these results.

The standard example of a very bad algorithm with the same competitive ratio as LRU and FIFO is FWF, which is shown to be strictly worse than both LRU and FIFO on any access graph (containing a path of length at least $k+1$), according to relative worst order analysis. Using relative worst order analysis, one can often obtain more nuanced results. This is also the case here for general access graphs, where we establish an incomparability result.

None of the algorithms we consider require prior knowledge of the underlying access graph. This issue was pointed out in [15] and [16] in connection with the limitations of some of the access graph results given in [5, 14, 18] and the Markov paging analogs in [21].

As relative worst order analysis is getting more established as a method for analyzing online algorithms, it is getting increasingly important that the the-

oretical toolbox is extended to match the options available when carrying out competitive analysis. Recently, in [13], list factoring [1, 3] was added as an analytical tool when using relative worst order analysis on list accessing problems [22, 2], and here we demonstrate that access graphs can also be included.

After a preliminary section, we prove that LRU is never worse than FIFO on paths or cycles. Then we establish separation results, showing that LRU is strictly better than FIFO on paths and cycles of length at least $k + 1$ and that both algorithms are strictly better than FWF on any graph containing a path of length at least $k+1$. The last result proves the incomparability of LRU and FIFO on general access graphs, using a family of graphs where the size is proportional to the length of the request sequence. We conclude with some open problems regarding determining completely for which classes of graphs LRU is better than FIFO. Some of the proofs have been omitted due to space constraints. They can be found in the full version [8].

2 Preliminaries

The *paging problem* is the problem of processing a sequence of page requests with the aim of minimizing the number of page faults. Pages reside in a large memory of size N, but whenever a page is requested, it must also be in the smaller cache of size $k < N$. If it is already present, we refer to this as a *hit*. Otherwise, we have a *fault* and must bring the page into cache. Except for start-up situations with a cache that is not full, this implies that some page currently in cache must be chosen to be evicted by a paging algorithm.

If $\mathbb{A}$ is a paging algorithm and I an input sequence, we let $\mathbb{A}(I)$ denote the number of faults that $\mathbb{A}$ incurs on I. This is also called the *cost* of $\mathbb{A}$ on I.

An important property of some paging algorithms that is used several times in this paper is the following:

Definition 1. *An online 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.*

The algorithms, Least-Recently-Used (LRU) and First-In/First-Out (FIFO) are examples of conservative algorithms. On a page fault, LRU evicts the least recently used page in cache and FIFO evicts the page which has been in cache the longest. Flush-When-Full (FWF), which is not conservative, evicts all pages in cache whenever there is a page fault and its cache is full.

An input sequence of page requests is denoted $I = \langle r_1, r_2, \ldots, r_{|I|} \rangle$. We use standard mathematical interval notation to denote subsequences. They can be open, closed, or semi-open, and are denoted by (r_a, r_b), $[r_a, r_b]$, $(r_a, r_b]$, or $[r_a, r_b)$. If S is a set of pages, we call a request interval *S-free* if the interval does not contain requests to any elements of S. We use the following notation for graphs.

Definition 2. *The path graph on N vertices is denoted P_N and a cycle graph on N vertices is denoted C_N. A walk is an ordered sequence of vertices where*

consecutive vertices are either identical or adjacent in the graph. A path is a walk in which every vertex appears at most once. The length of a walk $\mathcal{W}$ is the number of (not necessarily distinct) vertices in it, denoted by $|\mathcal{W}|$. The set of distinct vertices in a walk $\mathcal{W}$ is denoted by $\{\mathcal{W}\}$.

Definition 3. *An* access graph $G = (V, E)$ *is a graph whose vertex set corresponds to the set of pages that can be requested in a sequence. A sequence is said to* respect *an access graph, if the sequence of requests constitutes a walk in that access graph.*

In the relative worst order analyses carried out in this paper, permutations play a key role. We introduce some notation for this and then present the standard definition of the relative worst order quality measure.

For an algorithm $\mathbb{A}$, $\mathbb{A}_W(I)$ is the cost of the algorithm $\mathbb{A}$ on the worst reordering of the input sequence I, i.e., $\mathbb{A}_W(I) = \max_\sigma \mathbb{A}(\sigma(I))$, where σ is a permutation on $|I|$ elements and $\sigma(I)$ is a reordering of the sequence I.

Definition 4. *For any pair of paging algorithms $\mathbb{A}$ and $\mathbb{B}$, we define*

$$c_l(\mathbb{A}, \mathbb{B}) = \sup\{c \mid \exists b\colon\ \forall I\colon\ \mathbb{A}_W(I) \geq c\,\mathbb{B}_W(I) - b\} \text{ and}$$
$$c_u(\mathbb{A}, \mathbb{B}) = \inf\{c \mid \exists b\colon\ \forall I\colon\ \mathbb{A}_W(I) \leq c\,\mathbb{B}_W(I) + b\}$$

If $c_l(\mathbb{A}, \mathbb{B}) \geq 1$ or $c_u(\mathbb{A}, \mathbb{B}) \leq 1$, the algorithms are said to be comparable *and the* relative worst order ratio $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ *of algorithm $\mathbb{A}$ to $\mathbb{B}$ is defined. Otherwise, $\mathrm{WR}_{\mathbb{A},\mathbb{B}}$ is undefined. If $c_l(\mathbb{A}, \mathbb{B}) \geq 1$, then $\mathrm{WR}_{\mathbb{A},\mathbb{B}} = c_u(\mathbb{A}, \mathbb{B})$ and if $c_u(\mathbb{A}, \mathbb{B}) \leq 1$, then $\mathrm{WR}_{\mathbb{A},\mathbb{B}} = c_l(\mathbb{A}, \mathbb{B})$.*

If $\mathrm{WR}_{\mathbb{A},\mathbb{B}} < 1$, algorithms $\mathbb{A}$ and $\mathbb{B}$ are said to be comparable in $\mathbb{A}$'s favor. *Similarly, if $\mathrm{WR}_{\mathbb{A},\mathbb{B}} > 1$, the algorithms are said to be comparable in $\mathbb{B}$'s* favor.

When we use this measure to compare algorithms on a given access graph G, we use the notation $\mathbb{A}_W^G(I)$ to denote the cost of $\mathbb{A}$ on a worst permutation of I that respects G. Similarly, we use $\mathrm{WR}_{\mathbb{A},\mathbb{B}}^G$ to denote the relative worst order ratio of algorithms $\mathbb{A}$ and $\mathbb{B}$ on the access graph G.

Finally, let $\mathrm{Worst}(I, G, \mathbb{A})$ denote the set of worst orderings for the algorithm $\mathbb{A}$ of I respecting the access graph G, i.e., any sequence in $\mathrm{Worst}(I, G, \mathbb{A})$ is a permutation of I respecting G, and for any $I \in \mathrm{Worst}(I, G, \mathbb{A})$, $\mathbb{A}(I) = \mathbb{A}_W^G(I)$.

3 Paths

In [5, Theorem 13], it has been shown that if the access graph is a tree, then LRU is optimal among all online algorithms. Furthermore, in the case of path graphs, LRU matches the performance of an optimal offline algorithm.

Theorem 1. *For all I respecting the path P_N, $\mathrm{LRU}_W^{P_N}(I) \leq \mathrm{FIFO}_W^{P_N}(I)$.*

Proof. Consider any sequence I respecting P_N. Let I' be a worst ordering for LRU among the permutations of I respecting P_N. Then, using LRU's optimality on trees for the first inequality, $\mathrm{LRU}_W^{P_N}(I) = \mathrm{LRU}(I') \leq \mathrm{FIFO}(I') \leq \mathrm{FIFO}_W^{P_N}(I)$. □

4 Cycles

Almost this entire section is leading up to a proof that for all I respecting the access graph C_N, $\mathrm{LRU}_W^{C_N}(I) \leq \mathrm{FIFO}_W^{C_N}(I)$. Notice that this theorem is not trivial, since there exist sequences respecting the cycle access graph where FIFO does better than LRU. Consider, for example, the cycle on four vertices $C_4 = \langle 1, 2, 3, 4 \rangle$, $k = 3$, and the request sequence $I = \langle 2, 1, 2, 3, 4, 1 \rangle$. With this sequence, at the request to 4, LRU evicts 1 and FIFO evicts 2. Thus, FIFO does not fault on the last request and has one fault fewer than LRU. Note that on the reordering, $I' = \langle 1, 2, 2, 3, 4, 1 \rangle$, LRU still faults five times, but FIFO does too. This is the transformation which would be performed in Lemma 3 below, combined with the operation in the proof of Lemma 1 to reinsert requests which have been removed. Note that this is not a worst ordering for LRU, since LRU and FIFO both fault six times on $I'' = \langle 1, 2, 3, 4, 1, 2 \rangle$.

Each of the results leading up to the main theorem in this section is aimed at establishing a new property that we may assume in the rest of the section. Formally, these results state that if we can prove our end goal *with* the new assumption, then we can also prove it without. Thus, it is just a formally correct way of phrasing that we are reducing the problem to a simpler one. Some of the sequence transformations we perform in establishing these properties also remove requests, in addition to possibly reordering. The following general lemma allows us to do this in all of these specific cases.

Lemma 1. *Assume we are given an access graph G, a sequence I respecting G, and a sequence $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, G, \mathrm{LRU})$. We write I_{LRU} as the concatenation of three subsequences $\langle I_1, I_2, I_3 \rangle$. Let I' be $\langle I_1, I_2', I_3 \rangle$, where I_2' can be any subsequence (not necessarily of the same length as I_2) such that I' still respects G. Assume that* LRU *incurs at least as many faults on I_2' as on I_2, and the cache content, including information concerning which pages are least recently used, is exactly the same just after I_2' in I' as after I_2 in I_{LRU}. Assume further that I_2' is obtained from I_2 by removing some requests and/or reordering requests, and that $\{I\} = \{I'\}$. Then, $I' \in \mathrm{Worst}(I', G, \mathrm{LRU})$, and if $\mathrm{LRU}(I') \leq \mathrm{FIFO}_W^G(I')$, then $\mathrm{LRU}_W^G(I) \leq \mathrm{FIFO}_W^G(I)$.*

By repeatedly removing the $j-1$ hits in a sequence of j consecutive requests to the same page, we establish the following property:

Property 1. In proving for any access graph G, any sequence I respecting G, and any $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, G, \mathrm{LRU})$ that $\mathrm{LRU}(I_{\mathrm{LRU}}) \leq \mathrm{FIFO}_W^G(I)$, we may assume that I_{LRU} has no consecutive requests to the same page.

We give a collection of definitions enabling us to describe how a request sequence without consecutive requests to the same page behaves on the cycle.

Definition 5.

- *An arc is a connected component of a cycle graph. As a mathematical object, an arc is the same as a path (in this section), but refers to a portion of C_N, rather than a part of the walk defined by a request sequence.*

– *One can fix an orientation in a cycle so that the concepts of moving in a clockwise or anti-clockwise direction are well-defined. We refer to a walk as being* uni-directional *if each edge is traversed in the same direction as the previous, and abbreviate this* u-walk.
– *A request r_i in the request sequence is a* turn *if the direction changes at that vertex, i.e., if r_i is neither the first nor the last request and $r_{i-1} = r_{i+1}$. The vertex requested is referred to as a* turning point.
– *When convenient we will represent a request sequence I by its* turn sequence*,*

$$T = \langle A_1, v_1, A_2, v_2, \ldots, A_z, v_z \rangle,$$

where $T = I$, v_z is simply the last request of the sequence, all the other v_i's are the turns of the request sequence, and all the A_i's are u-walks. Thus, for all $i < z$, either $A_i \subseteq A_{i+1}$ or $A_{i+1} \subseteq A_i$. We refer to a turn v_i as a clockwise (anti-clockwise) turn *if the A_{i+1} goes in the clockwise (anti-clockwise) direction.*
– *Two turns are said to be* opposite *if they are in different directions.*
– *If for some $i < z$, $|A_{i+1} \cup \{v_{i+1}\}| \geq k$, then v_i is an* extreme turn*. Otherwise, v_i is a* trivial turn*.*

Most of the above is obvious terminology about directions around the circle. The last definition, on the other hand, is motivated by the behavior of the paging algorithms that we analyze. Not surprisingly, it turns out to be an important distinction whether or not the cache will start evicting pages before turning back. We treat this formally below.

We now reduce our problem to sequences without trivial turns.

Lemma 2. *Assume Property 1. For the access graph C_N, assume that for any I and $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, C_N, \mathrm{LRU})$, where I_{LRU} has no trivial turns, we have that $\mathrm{LRU}(I_{\mathrm{LRU}}) \leq \mathrm{FIFO}_W^{C_N}(I)$. Then, for any I, $\mathrm{LRU}_W^{C_N}(I) \leq \mathrm{FIFO}_W^{C_N}(I)$.*

We have now established the following property:

Property 2. We may assume that a worst ordering for LRU has no trivial turns.

Lemma 3. *Assume Properties 1–2. For the access graph C_N, assume that for any sequence I and $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, C_N, \mathrm{LRU})$, where I_{LRU} has turn sequence $\langle A_1, v_1, A_2, v_2, \ldots, A_z, v_z \rangle$ and $\forall i : \ |A_i| \geq k-1$, we have that $\mathrm{LRU}(I_{\mathrm{LRU}}) \leq \mathrm{FIFO}_W^{C_N}(I)$. Then, for any I, $\mathrm{LRU}_W^{C_N}(I) \leq \mathrm{FIFO}_W^{C_N}(I)$.*

Proof. By Property 2, we may assume that there are no trivial turns. Thus, we already know that for any i, $1 < i \leq z$, the result holds.

If $|A_1| < k-1$, then we replace I_{LRU} by $I'_{\mathrm{LRU}} = \langle v_1, A_2, \ldots \rangle$. This preserves the number of faults in the subsequence $\langle v_1, A_2 \rangle$ compared with $\langle A_1, v_1, A_2 \rangle$, and since $|A_2| \geq k-1$, LRU is in the same state after processing A_2 in I'_{LRU} as it was in processing I_{LRU}. By Lemma 1, we can use I'_{LRU}. □

We have now established the following property:

Property 3. We may assume that a worst ordering for LRU is of the form

$$\langle A_1, v_1, A_2, v_2, \ldots, A_z, v_z \rangle \text{ where } \forall i\colon\ |A_i| \geq k-1.$$

If the first three properties hold for some sequence, I, then it is easy to see that the number of turns determines how many hits LRU has on I.

Proposition 1. *If I has the form of Property 3 and contains no repeated requests to the same page, then* LRU *has exactly $(z-1)(k-1)$ hits on I.*

Next we show that we may assume that in a worst ordering for LRU, there is no turn which is followed by a full cycle in the opposite direction.

Definition 6. *Let u, v, and w be three distinct consecutive vertices on C_N. We refer to I as having an* overlap *if I can be written as $\langle \ldots, u, v, u, B, w, v, \ldots \rangle$. If I does not have an overlap, we refer to I as* overlap-free*.*

Lemma 4. *Assume Properties 1–3. For the access graph C_N, assume that for any I and $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, C_N, \mathrm{LRU})$, where I_{LRU} is overlap-free, we have that $\mathrm{LRU}(I_{\mathrm{LRU}}) \leq \mathrm{FIFO}_W^{C_N}(I)$. Then, for any I, $\mathrm{LRU}_W^{C_N}(I) \leq \mathrm{FIFO}_W^{C_N}(I)$.*

Proof. Let $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, C_N, \mathrm{LRU})$. If I_{LRU} has an overlap, we show that by reordering while respecting C_N an overlap-free sequence with at least as many faults can be constructed.

Assume that I_{LRU} has an overlap and consider a first occurrence of a vertex u in I_{LRU} such that I_{LRU} contains the pattern $\langle \ldots, u, v^1, u, B, w, v^2, \ldots \rangle$, where u, v, and w are consecutive vertices on C_N. The superscripts on v are just for reference, i.e., v^1 and v^2 are the same vertex.

We define $I' = \langle \ldots, u, v^1, w, B^R, u, v^2, \ldots \rangle$, where B^R denotes the walk B, reversed. Clearly, I' respects C_N. We now argue that I' incurs no more faults than I_{LRU}. Clearly, there is a turn at v^1 in I_{LRU}. If there is also a turn at v^2, then we have effectively just removed two turns. Then Proposition 1 implies that I_{LRU} cannot be a worst ordering. Thus, we can assume there is no turn at v^2.

In the transformation, we are removing the turn at v^1 and introducing one at v^2. Thus, since in the sequence I_{LRU} all u-walks between turns contained at least $k-1$ vertices, this is still the case after the transformation in I', except possibly for the u-walk from the newly created turn at v^2 to the next turn in the sequence. Let x denote such a next turn.

If the u-walk between v and x has at least $k-1$ vertices, then the transformed sequence has the same number of turns, all u-walks between turns contain at least $k-1$ vertices, and therefore I_{LRU} and I' have the same number of hits (and faults). In addition, the state of the caches after treating I_{LRU} up to x and I' up to x are the same.

If that u-walk contains fewer than $k-1$ vertices, we consider the next turn y after x. Since there are at least $k-1$ vertices in between x and y, we must pass v on the way to y.

Thus, consider $I_{\mathrm{LRU}} = \langle \ldots, u, v^1, u, B, w, v^2, B_1, x, B_2, v^3, B_3, y, \ldots \rangle$, having turns at v^1, x, and y, versus $I' = \langle \ldots, u, v^1, w, B^R, u, v^2, B_1, x, B_2, v^3, B_3, y, \ldots \rangle$, where there are turns at v^2, x, and y.

Comparing $\langle \ldots, u, v^1, u, B, w, v^2 \rangle$ with $\langle \ldots, u, v^1, w, B^R, u, v^2 \rangle$, both of them have least $k-1$ vertices on any u-walk between two turns, and the latter has one fewer turns. Thus, by Proposition 1, it has $k-1$ fewer hits.

By assumption, B_1 has fewer than $k-1$ vertices. Thus, comparing I_{LRU} and I' up to and including x, I' has at least as many faults.

In I_{LRU}, $\langle B_2, v^3 \rangle$ must all be hits, so up to and including v^3, I' has at least as many faults.

Since the u-walk leading to v^1 in I' contains at least $k-1$ vertices (not including v^1), and since the u-walk going from v^2 to y goes in the same direction, the requests in $\langle B_3, y \rangle$ must all be faults in I'.

Thus, we have shown that there are at least as many faults in I' as in I_{LRU}. In addition, the state of the caches after treating I_{LRU} up to y and I' up to y are the same.

With the transformation above, we do not incur more faults, and any first occurrence of a vertex u initiating an overlap pattern has been moved further towards the end of the sequence. Thus, we can apply this transformation technique repeatedly until no more such patterns exist. □

We have now established the following property:

Property 4. We may assume that a worst ordering is overlap-free.

Now we have all the necessary tools to prove the theorem of this section.

Theorem 2. *For all I respecting the cycle C_N,* $\mathrm{LRU}_W^{C_N}(I) \leq \mathrm{FIFO}_W^{C_N}(I)$.

Proof. We may assume Properties 1–4.

Consider any I and $I_{\mathrm{LRU}} \in \mathrm{Worst}(I, C_N, \mathrm{LRU})$. If there are no turns at all in I_{LRU}, both FIFO and LRU will fault on every request. If there is only one turn, FIFO will clearly fault as often as LRU on I_{LRU}, since we may assume that there is no overlap.

So, consider the first two turns v and v'. By Property 4, we cannot have the pattern $\langle \ldots, u, v, u, B, w, v, \ldots \rangle$. Thus, after the first turn, the edge from w to v can never be followed again. This holds symmetrically for v', which is a turn in the other direction. Thus, once the request sequence enters the arc between v and v', it can never leave it again. We refer to this arc as the *gap*. To be precise, since we are on a cycle, the gap is the arc that at the two ends has the neighbor vertices of v and v' from which edges to v and v', respectively, cannot be followed again, and such that v and v' are not part of the arc.

Assume without loss of generality that, after the first turn, if the request sequence enters the gap between v and v', then it does so coming from v'. After the first turn at v, the requests can be assumed to be on the path access graph P_N instead of the cycle C_N, where the access graph P_N starts with v, continues in the direction of the turn at v, and ends at the neighbor of v in the gap. In fact, we can assume that we are working on the access graph P_N from $k-1$ requests before the first turn at v, since all u-walks can be assumed to have at least that length. Let r_i be that request. Since there are no turns before v, starting with r_i, LRU and FIFO function as they would starting with an empty cache.

We divide $I_{\mathrm{LRU}} = \langle r_1, r_2, \ldots, r_{|I_{\mathrm{LRU}}|}\rangle$ up into the sequences $\langle r_1, r_2, \ldots, r_{i-1}\rangle$ and $\langle r_i, \ldots, r_{|I_{\mathrm{LRU}}|}\rangle$. The former is a u-walk, where LRU and FIFO both fault on every request, and the latter can be considered a request sequence on a path access graph as explained above, and the conclusion follows from Theorem 1. □

5 Separation on a path of length $k+1$

In the last sections, we showed that LRU was at least as good as FIFO on any path graph or cycle graph. Now we show that LRU is strictly better if these graphs contain paths of length at least $k+1$. We exhibit a family of sequences $\{I_n\}_{n\geq 1}$ such that $\mathrm{FIFO}_W^{P_N}(I_n) \geq \left(\frac{k+1}{2}\right)\cdot \mathrm{LRU}_W^{P_N}(I_n) + b$, for some fixed constant b, on path graphs P_N with $N \geq k+1$. Only $k+1$ different pages are requested in I_n. The same family of sequences is also used to show that FWF is worse than either LRU or FIFO. We number the vertices of the path graph P_N in order from 1 through N.

Theorem 3. *There exists a family of sequences, $I_n = \langle 1, \ldots, k, k+1, k, \ldots, 2\rangle^n$, respecting the access graph P_N, and a constant b such that the following holds:*

$$\lim_{n\to\infty} \mathrm{LRU}(I_n) = \infty, \text{ and for all } I_n, \ \mathrm{FIFO}_W^{P_N}(I_n) \geq \left(\frac{k+1}{2}\right)\cdot \mathrm{LRU}_W^{P_N}(I_n) + b.$$

We now have tight upper and lower bounds on the relative worst order ratio of FIFO to LRU on paths.

Theorem 4. *For any access graph G, if $\mathrm{WR}_{\mathrm{FIFO,LRU}}^{G} \geq 1$, then $\mathrm{WR}_{\mathrm{FIFO,LRU}}^{G} \leq \frac{k+1}{2}$. Thus, if $N \geq k+1$, then $\mathrm{WR}_{\mathrm{FIFO,LRU}}^{P_N} = \frac{k+1}{2}$.*

It was shown in [7] that for a complete graph, the relative worst order ratio of FWF to FIFO is exactly $\frac{2k}{k+1}$. This is also a lower bound for any graph containing P_{k+1}, but it is still open whether or not equality occurs in all sparser graphs.

Theorem 5. *For any access graph G which has a path of length at least $k+1$,*

$$\mathrm{WR}_{\mathrm{FWF,FIFO}}^{G} \geq \frac{2k}{k+1} \text{ and } \mathrm{WR}_{\mathrm{FWF,LRU}}^{G} = k.$$

6 Incomparability

In this section, we show that on some general classes of access graphs, LRU and FIFO are incomparable. We consider the cyclic access graph defined by the edge set $\{(1,2),(2,3),(3,4),(4,5),(5,1)\}$ and the request sequence $I_1 = \langle 1,5,1,2,3,4,5,1,2,1\rangle$ processed using a cache of size 4.

Lemma 5. *On any reordering of I_1 starting with* 1, LRU *incurs at least* 8 *faults and* FIFO *incurs at most* 7 *faults.*

Proof. It is trivial to check that LRU incurs 8 faults on I_1.

For FIFO, it is easy to check in the following that reorderings with repeated requests do not lead to more faults by FIFO. The reorderings of I_1 either have a prefix of the type $\{\langle 1, i, 1\rangle \mid i \in \{2, 5\}\}$ or $\{\langle 1, i, j\rangle \mid i \neq j \neq 1\}$. For the latter, examples being $\langle 1, 2, 3\rangle$ and $\langle 1, 5, 4\rangle$, the subsequence following the prefix contains 4 distinct pages. Since FIFO is conservative, it can incur at most 4 faults on that part after the prefix, bringing the total fault count up to at most 7.

The first four distinct page requests will always incur 4 faults, but for reorderings with the prefix $\{\langle 1, i, 1\rangle \mid i \in \{2, 5\}\}$, some pages are repeated within the first four requests. If the extended prefix is $\langle 1, i, 1, i\rangle$ for $i \in \{2, 5\}$, then the rest of the sequence still contains 4 distinct pages and again can add at most 4 faults to the previous 2, bringing the total up to at most 6. The only remaining case is a prefix of the form $\langle 1, i, 1, j\rangle$ where $i, j \in \{2, 5\}$, $i \neq j$. Here, there are 3 faults on the prefix. We divide the analysis of the rest of the sequence up into two cases depending on the next request following j:

For the first case, if the next request is 1, the extended prefix is $\langle 1, i, 1, j, 1\rangle$. However, then the next request to a page other than 1 is either to i or j and therefore not a fault. In addition, either there are no more i's or no more j's in the remaining part of the sequence, and again FIFO can then fault at most 4 times on this sequence with only 4 distinct pages.

For the second case, if the next request is $k \in \{3, 4\}$, then visiting $l \in \{4, 5 | l \neq k\}$ before the next j will give a prefix $\langle 1, i, 1, j, k, l\rangle$ with 5 faults, and the suffix must be $\langle i, 1, j, 1\rangle$ or $\langle i, 1, 1, j\rangle$, adding only one more fault. This gives 6 faults in total. If j is requested before l, the only possibilities are $\langle 1, 2, 1, 5, 4, 5, 1, 1, 2, 3\rangle$ and $\langle 1, 5, 1, 2, 3, 2, 1, 1, 5, 4\rangle$. In total, this gives only 5 faults. □

Note that the result above does not contradict our result about cycles. As predicted by that result, one of the worst orderings for LRU and FIFO would be $\langle 2, 1, 5, 4, 3, 2, 1, 5, 1, 1\rangle$, incurring 8 faults for both algorithms.

Using the cycle graph on which we processed I_1, we now construct a larger graph using "copies" of this graph as follows. For $2 \leq i \leq n$, we define I_i as a structural copy of I_1, i.e, we use new page names, but with the same relative order as in I_1 (like putting a "dash" on all pages in I_1). All these copies have their own set of pages such that no request in I_i appears in I_j for $i \neq j$. Just as I_1 implies a cycle graph that we denote X_1, so do each of these sequences and we let X_i denote the graph implied by I_i. Let $X_{i,k}$ denote the kth vertex in the ith copy and $I_{j,k}$ denote the kth request in the jth copy. To be precise, we define $I_i = \langle X_{i,1}, X_{i,5}, X_{i,1}, X_{i,2}, X_{i,3}, X_{i,4}, X_{i,5}, X_{i,1}, X_{i,2}, X_{i,1}\rangle$.

We define a graph $\mathcal{G}_n$ with a vertex set containing all $X_{i,k}$ and n additional vertices $u_1, u_2, \ldots, u_n$. Its edges are all the edges from the graphs X_i, $1 \leq i \leq n$, together with edges $(X_{i,1}, u_i)$ and $(u_i, X_{i+1,1})$ for all i, $1 \leq i \leq n-1$, plus the edge $(X_{n,1}, u_n)$.

Thus, $\mathcal{G}_n$ can be described as a chain of cycles, where each two neighboring cycles are separated by a single vertex. Clearly, the sequence $\mathcal{I}_n = \langle I_1, u_1, I_2, u_2, I_3, u_3, \ldots, I_n, u_n\rangle$ respects the access graph $\mathcal{G}_n$.

Theorem 6. *For the infinite family of sequences* $\{\mathcal{I}_n\}$ *respecting the access graph* $\mathcal{G}_n$*, the following two conditions hold:*

- $\lim_{n\to\infty} \mathrm{FIFO}(\mathcal{I}_n) = \infty$.
- *for all* $\mathcal{I}_n$, $\mathrm{LRU}_W^{\mathcal{G}_n}(\mathcal{I}_n) \geq \frac{9}{8} \cdot \mathrm{FIFO}_W^{\mathcal{G}_n}(\mathcal{I}_n)$.

Thus, although LRU is strictly better than FIFO on paths and cycles, FIFO is strictly better than LRU on the family of sequences, $\{\mathcal{I}_n\}$, respecting the family of graphs, $\{\mathcal{G}_n\}$. Note that $J_r = \langle X_{1,1}, X_{1,2}, X_{1,3}, X_{1,4}, X_{1,5}, X_{1,4}, X_{1,3}, X_{1,2}\rangle^r$, which only uses the first cycle of $\mathcal{G}_n$, trivially respects $\mathcal{G}_n$ for any r. By Theorem 3 for $k = 4$, $\mathrm{FIFO}_W^{\mathcal{G}_n}(J_r) \geq \left(\frac{k+1}{2}\right) \mathrm{LRU}_W^{\mathcal{G}_n}(J_r) - (k-1)$.

Thus, on the family $\{J_r\}$, LRU is better than FIFO. Combining with Theorem 6, we get:

Theorem 7. LRU *and* FIFO *are incomparable on the family of graphs* $\{\mathcal{G}_n\}$*, according to relative worst order analysis.*

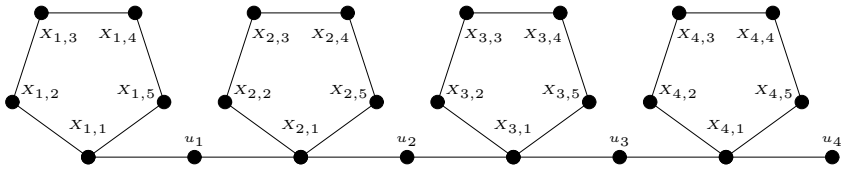

Fig. 1. The graph $\mathcal{G}_n$ for $n = 4$.

7 Open problems

We have determined that according to relative worst order analysis, LRU is better than FIFO on paths and cycles. On some classes of general access graphs, the two algorithms are incomparable. It would be interesting to get closer to determining exact access graphs classes characterizing relationships between the two algorithms. We believe that the results for paths and cycles will form fundamental building blocks in an attack on this problem. The most obvious class of access graphs to study next is trees. LRU can clearly do better than FIFO on any tree containing a path of length $k + 1$. We conjecture that LRU does at least as well as FIFO on any tree. One difficulty in establishing a proof of this is that for trees, as opposed to the cases of paths and cycles, there exist worst order sequences for LRU for which FIFO performs better than LRU.

For general access graphs, when showing that FIFO can do better than LRU, we used a family of access graphs, the size of which grew with the length of the input sequence. It would be interesting to know if this is necessary, or if such a separation result can be established on a single access graph of bounded size.

References

1. Albers, S., von Stengel, B., Werchner, R.: A combined BIT and TIMESTAMP algorithm for the list update problem. Inform. Proc. Letters 56, 135–139 (1995)
2. Albers, S., Westbrook, J.: Self-organizing data structures. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms — The State of the Art, LNCS, vol. 1442, pp. 13–51. Springer (1998)
3. Bentley, J.L., McGeoch, C.C.: Amortized analyses of self-organizing sequential search heuristics. Comm. ACM 28, 404–411 (1985)
4. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
5. Borodin, A., Irani, S., Raghavan, P., Schieber, B.: Competitive paging with locality of reference. Journal of Computer and System Sciences 50(2), 244–258 (1995)
6. Boyar, J., Favrholdt, L.M.: The relative worst order ratio for on-line algorithms. ACM Transactions on Algorithms 3(2) (2007), article No. 22
7. Boyar, J., Favrholdt, L.M., Larsen, K.S.: The relative worst order ratio applied to paging. Journal of Computer and System Sciences 73(5), 818–843 (2007)
8. Boyar, J., Gupta, S., Larsen, K.S.: Access graphs results for LRU versus FIFO under relative worst order analysis (2012), arXiv:1204.4047v1 [cs.DS]
9. Chrobak, M., Noga, J.: LRU is better than FIFO. Algorithmica 23(2), 180–185 (1999)
10. Denning, P.J.: The working set model for program behaviour. Comm. ACM 11(5), 323–333 (1968)
11. Denning, P.J.: Working sets past and present. IEEE Transactions on Software Engineering 6(1), 64–84 (1980)
12. Dorrigiv, R., López-Ortiz, A.: A survey of performance measures for on-line algorithms. SIGACT News 36(3), 67–81 (2005)
13. Ehmsen, M.R., Kohrt, J.S., Larsen, K.S.: List Factoring and Relative Worst Order Analysis. In: Jansen, K., Solis-Oba, R. (eds.) WAOA. LNCS, vol. 6534, pp. 118–129. Springer (2011)
14. Fiat, A., Karlin, A.R.: Randomized and multipointer paging with locality of reference. In: 27th Annual ACM Symposium on Theory of Computing. pp. 626–634 (1995)
15. Fiat, A., Mendel, M.: Truly online paging with locality of reference. In: 38th Annual Symposium on Foundations of Computer Science. pp. 326–335 (1997), extended version : CoRR, abs/cs/0601127, 2006
16. Fiat, A., Rosen, Z.: Experimental studies of access graph based heuristics: Beating the LRU standard? In: 8th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 63–72 (1997)
17. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Systems Tech. Journal 45(9), 1563–1581 (1966)
18. Irani, S., Karlin, A.R., Phillips, S.: Strongly competitive algorithms for paging with locality of reference. SIAM J. Comput. 25(3), 477–497 (1996)
19. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. Journal of the ACM 47(4), 617–643 (2000)
20. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. Algorithmica 3, 79–119 (1988)
21. Karlin, A.R., Phillips, S.J., Raghavan, P.: Markov paging. SIAM J. Comput. 30(3), 906–922 (2000)
22. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Comm. ACM 28(2), 202–208 (1985)