On the Separation and Equivalence of Paging Strategies

Spyros Angelopoulos* Reza Dorrigiv* Alejandro López-Ortiz*

Abstract

It has been experimentally observed that LRU and variants thereof are the preferred strategies for on-line paging. However, under most proposed performance measures for on-line algorithms the performance of LRU is the same as that of many other strategies which are inferior in practice. In this paper we first show that any performance measure which does not include a partition or implied distribution of the input sequences of a given length is unlikely to distinguish between any two lazy paging algorithms as their performance is identical in a very strong sense. This provides a theoretical justification for the use of a more refined measure. Building upon the ideas of concave analysis by Albers et al. [AFG05], we prove *strict* separation between LRU and all other paging strategies. That is, we show that LRU is the unique optimum strategy for paging under a deterministic model. This provides full theoretical backing to the empirical observation that LRU is preferable in practice.

1 Introduction

Paging is a fundamental problem in the context of the analysis of on-line algorithms. A paging algorithm mediates between a slower and a faster memory. Assuming a cache of size k, it decides which k memory pages to keep in the cache without the benefit of knowing in advance the *sequence* of upcoming page requests. After receiving the i-th page request and in the event the request results in a cache miss, the on-line algorithm must decide irrevocably which page to evict. The objective is to design efficient on-line algorithms in the sense that on a given request sequence the total cost, namely the total number of cache misses, is kept low.

Competitive analysis has long been established as the canonical approach for the analysis of on-line algorithms. The competitive ratio, first introduced formally by Sleator and Tarjan [ST85], has served as a practical measure for the study and classification of on-line algorithms. An algorithm (assuming a cost-minimization problem such as paging) is said to be α-competitive if the cost of serving any specific request sequence never exceeds α times the optimal cost (up to some additive constant) of an *off-line* algorithm which knows the entire sequence. The competitive ratio has been applied to a variety of problems and settings, mainly due to its amenability to analysis: the measure is relatively simple to define yet powerful enough to quantify, to a large extent, the performance of an on-line algorithm. On the other hand, there are known applications in which competitive analysis has yielded unsatisfactory results. Most notably, for the case of paging algorithms competitive analysis produces unrealistically pessimistic measures and fails to distinguish between algorithms which differ vastly in performance in practical settings, as first observed in [ST85]. Consider, for example, the paging strategies flush-when-full (FWF), least-recently-used (LRU) and first-in-first-out (FIFO). For the case of LRU and FIFO, Young established experimental values of the competitive ratio no larger than four [You94]. In contrast the competitive ratio for all three algorithms has a theoretical value of k. Furthermore, it has long been empirically established that LRU (and variants thereof) are, in practice, preferable paging strategies to all other known paging algorithms [SGG02]. An additional drawback of competitive analysis, as can easily be shown [BEY98], is that finite lookahead yields no improvement in the performance of an on-line algorithm. Once again, this is a rather counterintuitive conclusion: in practice, one expects that lookahead should improve performance, and a "reasonable" theoretical measure should reflect this reality.

Such anomalies have led to the introduction of many alternatives to the competitive analysis of on-line algorithms in general, and for the paging problem in particular (see [DLO05] for a comprehensive survey). In general, known alternative approaches rely on one or more of the following: i) defining a new measure as substitute of the competitive ratio; ii) limiting the power of the adversary; iii) employing different definitions for the concept of the "cost" of an algorithm; iv) incorporating certain assumptions concerning the request sequences. Competitive analysis uses an optimal off-line algorithm as a baseline to compare on-line algorithms. While this may be convenient, it is rather indirect: one could argue that in comparing A to B all we need to study is the relative cost of the algorithms on the request sequences. The approach we follow in this paper stems

*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ont., N2L 3G1, Canada. {sangelop, rdorrigiv, alopez-o}@uwaterloo.ca.

from this basic observation. Our definition focuses not on a specific worst case request sequence, but rather in the performance of the two algorithms on *all* possible sequences. We provide a formal definition of this intuitive observation, which is based on bijective mappings of the set of all possible sequences of a given length onto itself. We term this technique *Bijective Analysis* (c.f. Section 3).

Our results We begin by showing that a very large class of natural paging strategies known as *lazy* algorithms are in fact strongly equivalent under this rather strict bijective measure. In contrast, we show that LRU is strictly better than FWF (note that the latter is not a lazy strategy). Both of these results describe natural, "to-be-expected" properties of the corresponding paging strategies which competitive analysis nevertheless fails to yield. The strong equivalence of lazy algorithms is at first sight a negative result, however it can also be interpreted as evidence of an inherent difficulty to separate algorithms in any very general setting. In fact, it implies that to obtain theoretical separation between algorithms we must either induce a partition of the request sequence space (e.g. as in Albers et al. [AFG05]) or assume a distribution on the sequence space (e.g. as in Koutsoupias and Papadimitriou [KP00], Young [You98] and Becchetti [Bec04]). The latter group of approaches use probabilistic assumptions on the sequence space. However, since we are interested in separating algorithms under a deterministic model, we adopt concave analysis as introduced by Albers et al. which we then apply in the context of Bijective Analysis. Using this approach, we show formally the main result: namely that LRU is never outperformed in any possible subpartition on the request sequence space induced by concave analysis (c.f. Corollary 5.1), while it always outperforms *any other paging algorithm* in at least one subpartition of the sequence space (c.f. Theorem 5.2). This result proves separation between LRU and all other algorithms and provides theoretical backing to the observation that LRU is preferable in practice. Additionally, we provide concrete evidence that lookahead is beneficial: we show that LRU with lookahead as small as one (namely the sequence is revealed to the algorithm as consecutive pairs of requests) is strictly better than LRU without any lookahead.

Structure of paper In Section 2 we describe related work and introduce some standard definitions. In Section 3 we define formally the notion of Bijective Analysis and show strong equivalence between all lazy marking algorithms. These results formalize ideas that while perhaps familiar to many researchers of on-line problems had yet to be proved in a rigorous manner. We also show that LRU is strictly better than FWF under this measure. In Section 4 we show that Bijective Analysis also captures the effects of lookahead. This is in contrast to the competitive ratio under which an algorithm with lookahead performs no better than a similar algorithm without lookahead. In Section 5 we present our main result, i.e., separation between LRU and all other paging strategies.

2 Preliminaries and Related Work

In this section we overview some alternatives to the competitive ratio. We refer the reader to the survey of Dorrigiv and López-Ortiz [DLO05] for a more comprehensive and detailed exposition.

Loose competitiveness, which was first proposed by Young in [You94] and later refined in [You02], considers an off-line adversary that is oblivious to the parameter k (the cache size). The adversary must produce a sequence that is bad for most values of k rather than for just a specific value. It also ignores the sequences on which the on-line algorithm incurs a cost less than a certain threshold. This results in a weaker adversary and hence in paging algorithms with constant performance ratios. The *diffuse adversary* model by Koutsoupias and Papadimitriou [KP00] as well as Young [You98, You00] refines the competitive ratio by restricting the set of legal request sequences to those derived from a class (family) of probability distributions. This restriction follows from the observation that although a good performance measure could in fact use the actual distribution over the request sequences, determining the exact distribution of real-life phenomena is a difficult task (e.g. depending on the intended application different distributions might arise). The *Max/Max ratio*, introduced by Borodin and Ben-David [BDB94] compares on-line algorithms based on their amortized worst-case behaviour (here the amortization arises by dividing the cost of the algorithm over the length of the request sequence). The *relative worst order ratio* [BF03, BFL05, BM04] combines some of the desirable properties of the Max/Max ratio and the *random order ratio* (introduced in [Ken96] in the context of the on-line bin packing problem). As with the Max/Max ratio, it allows for direct comparison of two on-line algorithms. Informally, for a given request sequence the measure considers the worst-case ordering (permutation) of the sequence, for each of the two algorithms, and compares their behaviour on these orderings. It then finds among all possible sequences the one that maximizes this worst-case performance.

There are several measures that model paging with locality of reference. Borodin, Raghavan, Irani, and Schieber [BIRS95] proposed the *access graph* model in which the universe of possible request sequences is reduced to reflect that the actual sequences that can arise

depend heavily on the structure of the program being executed. The space of request sequences can then be modeled by a graph in which paths between vertices correspond to actual sequences. In a generalization of the access graph model, Karlin, Phillips, and Raghavan [KPR00] proposed a model in which the request sequences are distributed according to a Markov chain process. Becchetti [Bec04] refined the diffuse adversary model of Koutsoupias and Papadimitriou by considering only probabilistic distributions in which temporal locality of reference is present. Torng [Tor98] considered the decomposition of input sequences to phases in the same manner as marking algorithms. He then modeled locality of reference by restricting the input to sequences with long average phase length. Using the *full access cost model*, he computed the performance of several paging algorithms on sequences with high locality of reference.

Most notably, Albers, Favrholdt, and Giel [AFG05] introduced a model in which input sequences are classified according to a measure of locality of reference. The measure is based on Denning's working set concept [Den68] which is supported by extensive experimental results. The technique used, which we term *concave analysis*, reflects the fact that efficient algorithms must perform competitively in each class of inputs of similar locality of reference, as opposed to the worst case alone. It should be noted that [AFG05] focuses on the *fault rate* as the measure of the cost of an algorithm, as opposed to the traditional definition of cost as the number of cache misses. Recently, Panagiotou and Souza proposed a model that explains the good performance of LRU in practice [PS06]. They classify input sequences according to some parameters and prove an upper bound on the competitive ratio of LRU as a function of these parameters. Then they argue that sequences in practice have parameters that lead to constant competitive ratio for LRU. Table 1 summarizes the separation results achieved by the alternative measures presented.

3 Bijective Analysis and Average Analysis

Most alternative measures to the competitive ratio usually restrict the set of legal input sequences or consider models other than the standard off-line adversary to compare the behaviour of an on-line algorithm [DLO05]. In this paper, we use both techniques: more precisely, we use the model for paging with locality of reference of Albers et al. [AFG05] combined with a refined comparison model. For an on-line algorithm $\mathcal{A}$ and an input sequence σ, let $\mathcal{A}(\sigma)$ be the cost incurred by $\mathcal{A}$ on σ. Denote by $\mathcal{I}_n$ the set of all input sequences of length n. The first comparison model we introduce is *Bijective Analysis*. In this model, we aim to pair input sequences for $\mathcal{A}$ and $\mathcal{B}$ using a bijection in such a way that the cost of $\mathcal{A}$ on sequence σ is no more than the cost of $\mathcal{B}$ on the image of σ.

DEFINITION 3.1. *We say that an on-line algorithm $\mathcal{A}$ is* no worse *than an on-line algorithm $\mathcal{B}$ according to Bijective Analysis if there exists an integer $n_0 \geq 1$ such that for each $n \geq n_0$, there is a bijection $b : \mathcal{I}_n \leftrightarrow \mathcal{I}_n$ satisfying $\mathcal{A}(\sigma) \leq \mathcal{B}(b(\sigma))$ for each $\sigma \in \mathcal{I}_n$. We denote this by $\mathcal{A} \preceq_b \mathcal{B}$. Otherwise we denote the situation by $\mathcal{A} \not\preceq_b \mathcal{B}$. Similarly, we say that $\mathcal{A}$ and $\mathcal{B}$ are* the same *according to Bijective Analysis if $\mathcal{A} \preceq_b \mathcal{B}$ and $B \preceq_b A$. This is denoted by $\mathcal{A} \equiv_b \mathcal{B}$. Finally we say $\mathcal{A}$ is* better *than $\mathcal{B}$ according to Bijective Analysis if $\mathcal{A} \preceq_b \mathcal{B}$ and $\mathcal{B} \not\preceq_b \mathcal{A}$. We denote this by $\mathcal{A} \prec_b \mathcal{B}$.*

Observe that similar to Max/Max ratio, this measure considers sequences of the same length and compares two on-line algorithms directly. However it compares *all* sequences in $\mathcal{I}_n$ rather than only the worst sequence. A related comparison model can be obtained by considering the average number of faults that a paging algorithm incurs on sequences of a certain length.

DEFINITION 3.2. *We say that an on-line algorithm $\mathcal{A}$ is* no worse *than an on-line algorithm $\mathcal{B}$ according to Average Analysis if there exists an integer $n_0 \geq 1$ such that for each $n \geq n_0$, $\sum_{I \in \mathcal{I}_n} \mathcal{A}(I) \leq \sum_{I \in \mathcal{I}_n} \mathcal{B}(I)$. We denote this by $\mathcal{A} \preceq_a \mathcal{B}$. Otherwise we denote the situation by $\mathcal{A} \not\preceq_a \mathcal{B}$. $\mathcal{A} \equiv_a \mathcal{B}$, and $\mathcal{A} \prec_a \mathcal{B}$ are defined as for Bijective Analysis.*

OBSERVATION 3.1. *If $\mathcal{A} \not\preceq_a \mathcal{B}$, then $\mathcal{A} \not\preceq_b \mathcal{B}$. As well, if $\mathcal{A} \preceq_b \mathcal{B}$, then $\mathcal{A} \preceq_a \mathcal{B}$ with similar statements holding for $\equiv_b$ and $\prec_b$.*

Example. We use a simple example to contrast these models. Let $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ be three on-line algorithms and $\mathcal{I}_n = \{\sigma_1, \sigma_2, \cdots, \sigma_{10}\}$ be the set of all possible input sequences of length n. Suppose that the cost of $\mathcal{A}$, $\mathcal{B}$, C, and the optimal off-line algorithm OPT on input sequences are as follows:

σ	$\mathcal{A}(\sigma)$	$\mathcal{B}(\sigma)$	$\mathcal{C}(\sigma)$	$OPT(\sigma)$
σ_1	5	6	6	3
σ_2	7	8	8	2
σ_3	3	4	3	4
σ_4	9	4	3	1
σ_5	7	10	8	3
σ_6	5	7	6	4
σ_7	3	6	4	2
σ_8	7	6	7	5
σ_9	5	8	5	2
σ_{10}	7	10	9	3

Measure	Positive results	Negative results	Comments
Competitive ratio [ST85]	amenable to analysis, applied extensively to a multitude of settings	LRU=FIFO=FWF	canonical on-line model
Loose competitiveness [You94]	more realistic ratios for paging algorithms		
Adequate Performance Measure [PS06]	more realistic ratios for paging algorithms		
Diffuse adversary [KP00, You00, Bec04]	more realistic ratios for paging algorithms, LRU < FWF, LRU < FIFO		probabilistic analysis
Access Graph [BIRS95]	LRU≤FIFO		
Max/Max ratio [BDB94]	direct comparison, influence of lookahead	LRU=FIFO=FWF	
Relative worst order ratio [BFL05]	direct comparison, influence of lookahead, LRU < FWF	LRU=FIFO	
Concave analysis [AFG05]	LRU < FIFO, FWF		uses the fault rate cost model
Torng's model [Tor98]	more realistic ratios, influence of lookahead	LRU=FIFO=FWF	uses full access cost model
Bijective Analysis [**this paper**]	direct comparison, influence of lookahead, FWF < LRU	LRU=FIFO	
Average Analysis+ Concave analysis [**this paper**]	LRU is the sole best paging algorithm		

Table 1: Separation results under alternative measures for analysis of on-line paging algorithms.

We have $\sum_\sigma \mathcal{A}(\sigma) = 58$, $\sum_\sigma \mathcal{B}(\sigma) = 69$, and $\sum_\sigma \mathcal{C}(\sigma) = 59$. Therefore $\mathcal{A} \prec_a \mathcal{B}$, $\mathcal{A} \prec_a \mathcal{C}$, and $\mathcal{C} \prec_a \mathcal{B}$. We have $\mathcal{B} \not\preceq_b \mathcal{A}$, because $\mathcal{B} \not\preceq_a \mathcal{A}$. We also have $\mathcal{A} \preceq_b \mathcal{B}$ by considering the bijection that maps $\sigma_1, \sigma_2, \ldots, \sigma_{10}$ to σ_1, σ_2, σ_3, σ_5, σ_6, σ_7, σ_4, σ_9, σ_8, and σ_{10}, respectively. Therefore $\mathcal{A}$ is better than $\mathcal{B}$ according to Bijective Analysis, i.e., $\mathcal{A} \prec_b \mathcal{B}$. Although $\mathcal{A}$ is better than $\mathcal{C}$ according to Average Analysis, the two algorithms are not comparable according to Bijective Analysis. Since $\mathcal{A} \prec_a \mathcal{C}$, we conclude that $\mathcal{C} \not\preceq_b \mathcal{A}$. Also we have $\mathcal{A} \not\preceq_b \mathcal{C}$ because $\mathcal{C}$ incurs a cost less than 5 on 3 sequences while $\mathcal{A}$ incurs a cost less than 5 only on 2 sequences. Now consider the competitive ratio of these algorithms. The competitive ratio of $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ is 9, 4, and 4 respectively. Although $\mathcal{A}$ seems to have better overall performance than $\mathcal{B}$ and $\mathcal{C}$, its bad performance on a single sequence, σ_4, leads to its very bad competitive ratio.

Separation between certain Paging Algorithms. As a warm-up, we will first show that LRU is better than FWF according to Bijective Analysis (recall that these algorithms have the same competitive ratio). We also prove that all lazy algorithms are equivalent according to Bijective Analysis. For the remainder of this section let N denote the number of pages in slow memory.

LEMMA 3.1. *$LRU \preceq_b FWF$.*

Proof. We prove that for every $n \geq 1$ there is a bijection $b^n : \mathcal{I}_n \leftrightarrow \mathcal{I}_n$ so that $LRU(\sigma) \leq FWF(b^n(\sigma))$ for each $\sigma \in I_n$. We show this by induction on n, the length of input sequences. For $n \leq k$, this is trivial as each sequence has at most k distinct pages and LRU and FWF behave the same. Assume that it is true for all $n \leq h$, where $h \geq k$. We prove that it is also true for $n = h + 1$. We define a new bijection $b^{h+1} : \mathcal{I}_{h+1} \leftrightarrow \mathcal{I}_{h+1}$, which maps the continuations of a sequence σ to the continuations of the sequence $b^h(\sigma)$ in the image and more specifically, maps LRU's last-fault continuation sequences of σ into FWF's last-fault continuation sequences of $b^h(\sigma)$.

First assume that σ has at least k distinct pages. Then LRU's cache contains k pages after serving σ. Therefore, there are k last-hit sequences and $N-k$ last-fault sequences in the continuation of σ for LRU. FWF's cache contains at most k pages after serving $b^h(\sigma)$. Thus there are at least $N-k$ last-fault sequences for FWF in the continuation of σ.

Alternatively, if σ has $k' < k$ distinct pages then from our construction $b(\sigma)$ also contains exactly k' distinct pages and the number of last-fault and last-

hit continuations for each algorithm match. Hence in either case the number of last-fault sequences in LRU is no larger than the number of last-fault sequences for FWF and we can define an injective mapping b^{h+1} from the former into the latter. We then arbitrarily map the remaining (last-hit) LRU continuation sequences of σ to the remaining unused sequences in the continuation of $b^h(\sigma)$. Clearly from the construction the bijection maps a request sequence in LRU to a request sequence of FWF with the same or more number of page faults, as claimed. □

LEMMA 3.2. *FWF $\not\preceq_b$ LRU.*

Proof. We prove this by contradiction. Assume that we have $FWF \preceq_b LRU$ and so there is an $n_0 \geq 1$ so that for each $n \geq n_0$ we have the bijection $b^n : \mathcal{I}_n \leftrightarrow \mathcal{I}_n$. Recall that we can partition a sequence into a number of consecutive phases so that each phase contains exactly k distinct pages. LRU incurs at most k faults in each phase. FWF empties its cache at the end of each phase and incurs exactly k faults in each phase. Therefore we have $LRU(\sigma) \leq FWF(\sigma)$ for each sequence σ. Thus the desired bijection exists only if we have $FWF(\sigma) = LRU(\sigma)$ for every sequence of length $n \geq n_0$. Consider a sequence $\sigma = p_1 p_2 \dots p_h$ $(h \geq n_0)$ so that σ contains at least k distinct pages and p_h is the first page of a phase. Therefore p_h causes FWF to flush its cache, which now contains only one page after serving σ. Now consider the set of continuations of σ. The number of last-fault sequences among these for LRU and FWF is $N - k$ and $N - 1$, respectively. Therefore there are at least $k - 1$ sequences for every σ for which the cost of LRU is strictly less than the cost of FWF and hence a bijection as required does not exist. □

Combining Lemma 3.1 and Lemma 3.2 we obtain strict separation of the performance of LRU and FWF.

THEOREM 3.1. *LRU $\prec_b$ FWF.*

Note that we can use exactly the same argument as in Lemma 3.1 to prove the following theorems.

THEOREM 3.2. *FIFO $\preceq_b$ FWF, FIFO $\preceq_b$ LRU, and LRU $\preceq_b$ FIFO. Thus we have LRU $\equiv_b$ FIFO.*

THEOREM 3.3. *Let $\mathcal{A}$ and $\mathcal{B}$ be two arbitrary lazy algorithms. Then we have $\mathcal{A} \equiv_b \mathcal{B}$.*

These results explain why most measures are unable to separate lazy algorithms such as LRU and FIFO. This supports the fact that we must constrain sequences to the ones most often appearing in practice such as those with locality of reference so that we may distinguish differing paging algorithms.

4 Influence of Lookahead

In this section we demonstrate that Bijective Analysis reflects the effects of lookahead. Let LRU(ℓ) be a modification of LRU defined for a lookahead of size ℓ as follows [Alb97]. On a fault, LRU(ℓ) evicts a page in the cache that is least recently used among the pages that are not in the current lookahead. Our model reflects the influence of the lookahead in that LRU(ℓ) $\prec_b$ LRU, i.e. LRU(ℓ) is better than LRU according to Bijective Analysis. Intuitively, LRU(ℓ) $\preceq_b$ LRU, because LRU(ℓ) behaves pretty much the same as LRU except when it knows it can do better.

LEMMA 4.1. *The cost of LRU(ℓ) is no more than the cost of LRU on any given sequence of requests, hence LRU(ℓ) $\preceq_b$ LRU.*

Proof. Assume for the sake of contradiction that there is a sequence $\sigma = p_1 p_2 \dots p_m$ on which LRU(ℓ) incurs strictly more faults than LRU. Let a be the smallest index so that p_a is a hit for LRU and a fault for LRU(ℓ). Suppose that the most recent eviction of p_a by LRU(ℓ) is at time r on the request p_r. Therefore we have $p_i \neq p_a$ for $r \leq i \leq a$ and furthermore LRU does not evict p_a at any time t, where $r \leq t \leq a$. Let $p_{r_1}, p_{r_2}, \dots, p_{r_k}$ be the pages in LRU's cache at time r so that p_{r_i} is less recently used than p_{r_j} at time r if and only if $i < j$. Note that since a is the smallest index so that p_a is a hit for LRU and a fault for LRU(ℓ), LRU incurs a fault on p_r and evicts p_{r_1}. Suppose that $p_{r_x} = p_a$ for some $1 < x \leq k$ and let $\mathcal{L}_r$ the set of pages in the lookahead of size ℓ at time r. We consider two cases:

Case 1: All the pages $p_{r_1}, p_{r_2}, \dots, p_{r_{x-1}}$ are in LRU(ℓ)'s cache at time r. Since LRU(ℓ) evicts $p_{r_x} = p_a$ at time r, we should have $p_{r_i} \in \mathcal{L}_r$ for $1 \leq i \leq x - 1$. Let y be the largest index so that $r \leq y \leq r + \ell$, $p_y \in \{p_{r_1}, p_{r_2}, \dots, p_{r_{x-1}}\}$, and LRU incurs a fault at time y. Note that since $p_{r_1} \in \mathcal{L}$ and LRU evicts p_{r_1} at time r, y exists. We claim that LRU should evict $p_{r_x} = p_a$ at time y. Since p_a has not been requested between times r and y, the only pages that can be less recently used than p_a are $p_{r_1}, p_{r_2}, \dots, p_{r_{x-1}}$. Assume that at time y, LRU evicts the page p_{r_z} for some $1 \leq z \leq x - 1$. p_{r_z} should not be requested between times r and y; otherwise p_a is less recently used than it. We know that $p_{r_z} \in \mathcal{L}_r$ and therefore p_{r_z} will be requested at least once between the times $y+1$ and $r+\ell$. The first such request is a fault on a page (p_{r_z}) that is in $\{p_{r_1}, p_{r_2}, \dots, p_{r_{x-1}}\}$; this contradicts the choice of y. Therefore p_a is the least recently used page for LRU at time y and LRU evicts it. This contradicts the fact that LRU does not evict p_a on a fault at any time $r \leq t \leq a$.

Case 2: There is a page $p \in \{p_{r_1}, p_{r_2}, \ldots, p_{r_{x-1}}\}$ that is not in LRU(ℓ)'s cache at time r. Let $r' < r$ be the last time that LRU(ℓ) has evicted p. Since p is in LRU's cache and not in LRU(ℓ)'s cache at time r, we have $p_i \neq p$ for $r' \leq i \leq r-1$ and furthermore LRU does not evict p at any time t, where $r' \leq t \leq r-1$. This reduces to the situation discussed at the beginning of this proof, with $a = r$ and $r = r'$. Since a is a finite number and we strictly decrease our time of interest, after a finite number of applications of case 2 this reduces to case 1.

Thus we cannot have any request on which LRU(ℓ) incurs a fault and LRU does not, and hence LRU(ℓ) does not incur more faults than LRU on any sequence. □

LEMMA 4.2. *There exists a sequence in which LRU(ℓ) outperforms LRU, therefore LRU $\not\preceq_b$ LRU(ℓ).*

Proof. From Lemma 4.1, we know that LRU has the same or higher number of page faults as LRU(ℓ) on each sequence of length at least n_0. So it suffices to show that on at least one sequence LRU(ℓ) strictly outperforms LRU. Consider a sequence σ of size $n_1 \geq n_0$ which contains several consecutive copies of the subsequence $p_1 p_2 \ldots p_k p_{k+1}$. LRU incurs a fault on all pages of σ and therefore the cost of LRU on σ is n_1. The cost of LRU(ℓ) on the other hand is $n_1/(\ell+1)$. □

THEOREM 4.1. *LRU(ℓ) $\prec_b$ LRU.*

5 Paging with Locality of Reference

In Section 3, we proved that all lazy algorithms are strongly equivalent according to Bijective Analysis. However, this analysis ignored that in practice request sequences exhibit *locality of reference*. We follow the model of Albers et al. [AFG05], in which a request sequence has high locality of reference if the number of distinct pages in a window of size n is small. Consider a function that represents the maximum (or average) number of distinct pages in a window of size n, in a request sequence. Extensive experiments with real data show that this function can be bounded by a concave function for most practical request sequences [AFG05]. Let f be an increasing concave function. There are two possible ways to model the locality. In the *Max-Model* we say that a request sequence is *consistent* with f if the number of distinct pages in any window of size n is at most $f(n)$, for any $n \in \mathcal{N}$. In the *Average-Model* we say that a request sequence is consistent with f if the average number of distinct pages in a window of size n is at most $f(n)$, for any $n \in \mathcal{N}$. Now we can model locality by considering only those request sequences that are consistent with f. Albers et al. consider a slightly more restrictive class of functions called concave* functions.

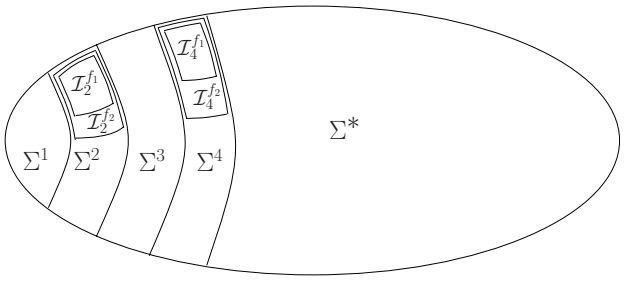

Figure 1: Partition of the input space induced by different choices of f.

DEFINITION 5.1. *[AFG05] A function $f : N \to R_+$ is concave* if*

1. $f(1) = 1$ *and*
2. $\forall n \in \mathcal{N} : f(n+1) - f(n) \geq f(n+2) - f(n+1)$.

In the Max-Model we additionally require that f be surjective on the integers between 1 and its maximum value.

We restrict the input sequences to those consistent with a given concave* function f in the Max-Model. Let $\mathcal{I}^f$ denote the set of such sequences. We can easily modify the definitions of Bijective Analysis and Average Analysis (Definition 3.1 and Definition 3.2) by considering $\mathcal{I}^f$ instead of $\mathcal{I}$. We denote the corresponding relations by $\mathcal{A} \preceq_b^f \mathcal{B}$, $\mathcal{A} \preceq_a^f \mathcal{B}$, etc. Note that we can make any sequence consistent with f by repeating every request a sufficient number of times. Therefore even if we restrict the input to sequences with high locality of reference, there is a worst case sequence for LRU that is consistent with f and therefore the competitive ratio of LRU is the same as in the standard model [1]. Figure 1 shows the partition of the input space induced by the choice of f. Observe that the performance of a paging algorithm is now evaluated within the subset of request sequences of a given length whose locality of reference is consistent with f, i.e. $\mathcal{I}_n^f$.

Note that the inductive argument used to prove that all lazy algorithms are equivalent according to Bijective Analysis does not necessarily carry through under concave analysis. When a continuation is proposed we must ensure that it remains within the set $\mathcal{I}^f$ to be valid.

Consider a fixed concave* function f. Let $\mathcal{I}_n^f$ denote sequences of length n in $\mathcal{I}^f$ and $\mathcal{A}$ be an arbitrary paging algorithm. We call a sequence *bad* for $\mathcal{A}$ if $\mathcal{A}$ incurs a fault on its last request; otherwise we call it a *good* sequence for $\mathcal{A}$. Let $B_h(\mathcal{A})$ be the number of

[1] This is one of the reasons that Albers et al. [AFG05] use the fault rate as a measure of performance.

sequences in $\mathcal{I}_h^f$ that are bad for $\mathcal{A}$. For a sequence $\sigma \in \mathcal{I}_h^f$, let $B_{h+1}(\mathcal{A} \mid \sigma)$ denote the number of sequences in $\mathcal{I}_{h+1}^f$ that have σ as their prefix and are bad for $\mathcal{A}$. Define $G_h(\mathcal{A})$ and $G_{h+1}(\mathcal{A} \mid \sigma)$ in an analogous way for good sequences. Intuitively, a good algorithm maintains its good sequences in the set of sequences with high locality of reference and hence can safely perform the continuation. Observe that LRU naturally fits this criterion: the most recently accessed sequences are exactly those that are in its cache, and therefore good (i.e. last-hit) sequences for LRU are more likely to be in sequences with high locality of reference. We formalize this intuition in the rest of this section.

LEMMA 5.1. *For any integer $h > 0$ and any paging algorithm $\mathcal{A}$, $B_h(LRU) \leq B_h(\mathcal{A})$.*

Proof. If $h = 1$, then every sequence of $\mathcal{I}_h$ is consistent with f and each algorithm incurs a fault on its last request (recall that algorithms start with an empty cache). Therefore we have $B_1(LRU) = |\mathcal{I}_1^f| = N = B_1(\mathcal{A})$. If $h > 1$, consider an arbitrary sequence $\sigma \in \mathcal{I}_{h-1}^f$. If σ has at most k distinct pages, then LRU and $\mathcal{A}$ have the same pages in their cache after serving σ and therefore $B_h(LRU \mid \sigma) = B_h(\mathcal{A} \mid \sigma)$. Otherwise, both LRU and $\mathcal{A}$ have filled their cache with k pages after serving σ. The next page requested can be an arbitrary page, provided that the number of different request pages thus far does not surpass the limit imposed by f.

From the definition of f, repeating the last request of a sequence $\sigma \in \mathcal{I}_{h-1}^f$ is always consistent with f. Repeating the second to last sequence may or may not be consistent with f, however if the second to last sequence is not consistent neither is any other request. This implies that for every good request for A that is consistent with f, there is a good request for LRU that is consistent with f. Hence $G_h(LRU \mid \sigma) \geq G_h(\mathcal{A} \mid \sigma)$. Now since the good and bad continuations form a partition of the set of valid extensions of σ, the inequality above implies $B_h(LRU \mid \sigma) \leq B_h(\mathcal{A} \mid \sigma)$. To conclude observe that $B_h(X) = \sum_{\sigma \in \mathcal{I}_{h-1}} B_h(X \mid \sigma)$ for any algorithm X. Hence

$$\begin{aligned} B_h(LRU) &= \sum_{\sigma \in \mathcal{I}_{h-1}} B_h(LRU \mid \sigma) \\ &\leq \sum_{\sigma \in \mathcal{I}_{h-1}} B_h(\mathcal{A} \mid \sigma) = B_h(\mathcal{A}) \end{aligned}$$

as claimed. □

Lastly, we show that LRU strictly outperforms all other caching algorithms.

DEFINITION 5.2. *Let m be an integer, $\mathcal{A}$ and $\mathcal{B}$ be paging algorithms, and f be a concave* function. $\mathcal{A}$ is said to* (m, f)-dominate *$\mathcal{B}$ if we have $\sum_{\sigma \in \mathcal{I}_m^f} \mathcal{A}(\sigma) \leq \sum_{\sigma \in \mathcal{I}_m^f} \mathcal{B}(\sigma)$. $\mathcal{A}$ is said to* dominate *$\mathcal{B}$ if there exists an integer $m_0 \geq 1$ so that for each $m \geq m_0$ and every concave* function f, $\mathcal{A}$ (m, f)-dominates $\mathcal{B}$.*

OBSERVATION 5.1. *$\mathcal{A} \preceq_a \mathcal{B}$ if and only if there exists an integer $m_0 \geq 1$ so that $\mathcal{A}$ (m, f)-dominates $\mathcal{B}$ for each $m \geq m_0$.*

LEMMA 5.2. *For every paging algorithm $\mathcal{A}$, LRU dominates $\mathcal{A}$.*

Proof. Let f be an arbitrary concave* function and m be a positive integer. For any $1 \leq i \leq m$, let $\mathcal{F}_{i,m}(\mathcal{A})$ be the number of sequences in $\mathcal{I}_m^f$ for which $\mathcal{A}$ incurs a fault on the i^{th} request. We will show that $\mathcal{F}_{i,m}(LRU) \leq \mathcal{F}_{i,m}(\mathcal{A})$ for any $1 \leq i \leq m$. For $i = 1$, we have $\mathcal{F}_{1,m}(LRU) = \mathcal{F}_{1,m}(\mathcal{A}) = |\mathcal{I}_m^f|$. Now assume that $i > 1$. Let σ be an arbitrary sequence of length $i - 1$, and let T_σ denote the set of sequences in $\mathcal{I}_m^f$ that have σ as their prefix. Denote by $\mathcal{F}_{i,m}(\mathcal{A} \mid \sigma)$ the number of sequences in T_σ for which $\mathcal{A}$ incurs a fault on the i^{th} request.

If σ contains at most k distinct pages, then LRU and $\mathcal{A}$ behave the same on σ and we have $\mathcal{F}_{i,m}(LRU \mid \sigma) = \mathcal{F}_{i,m}(\mathcal{A} \mid \sigma)$. Assume then that σ has more than k distinct pages. We can partition T_σ into four subsets: (1) T_σ^1: sequences in which neither LRU nor $\mathcal{A}$ incur a fault on the i^{th} page request, (2) T_σ^2: sequences in which both LRU and $\mathcal{A}$ incur a fault on the i^{th} page request, (3) T_σ^3: sequences in which $\mathcal{A}$ incurs a fault on the i^{th} page request, but LRU does not. (4) T_σ^4: sequences in which LRU incurs a fault on the i^{th} page request, but $\mathcal{A}$ does not.

We have $\mathcal{F}_{i,m}(LRU \mid \sigma) = |T_\sigma^2| + |T_\sigma^4|$, and $\mathcal{F}_{i,m}(\mathcal{A} \mid \sigma) = |T_\sigma^2| + |T_\sigma^3|$. We show that $|T_\sigma^4| \leq |T_\sigma^3|$ by proving that there exists a one-to-one mapping d from T_σ^4 to T_σ^3. For $1 \leq q \leq 4$, let P_σ^q be the set of pages that are requested as the i^{th} page of a sequence in T_σ^q. If a page p appears as the i^{th} request of a sequence in $\mathcal{I}_i^f$, then it will appear as the i^{th} request of a sequence in $\mathcal{I}_m^f$, namely the one obtained by repeating p after the i^{th} request. Therefore, we have $B_i(LRU \mid \sigma) = |P_\sigma^2| + |P_\sigma^4|$ and $B_i(\mathcal{A} \mid \sigma) = |P_\sigma^2| + |P_\sigma^3|$. We know that $B_i(LRU \mid \sigma) \leq B_i(\mathcal{A} \mid \sigma)$ from the proof of Lemma 5.1; therefore $|P_\sigma^4| \leq |P_\sigma^3|$ and there is a one-to-one mapping r from P_σ^4 to P_σ^3. We use the mapping r to define the desired mapping d. Consider an arbitrary sequence $S = p_1 p_2 \dots p_m \in T_\sigma^4$. Let $p_i = x$ and $y = r(x)$. According to definitions we know that on the i^{th} request of a sequence in T_σ, x is a fault for LRU and a hit for $\mathcal{A}$, while y is a hit for LRU and a fault for $\mathcal{A}$. Let $\sigma_x \in \mathcal{I}_i^f$ be the sequence obtained by appending the page x to σ, and $\sigma_y \in \mathcal{I}_i^f$ be the sequence obtained by

appending the page y to σ. On serving σ_x, the last page (x) is a fault for LRU; therefore x is not among the last k distinct pages in σ. LRU does not incur a fault on the last page of σ_y; thus y is among the last k distinct pages of σ. Hence if starting from i^{th} request, we convert each x in a sequence in T_{σ_x} to y, we will obtain a sequence that is consistent with f, i.e., a sequence in T_{σ_y}. This gives us a one-to-one mapping from T_{σ_x} to T_{σ_y}. By a similar process for the pages in P^4_σ, we obtain a one-to-one mapping from T^4_σ to T^3_σ. Therefore

$$|T^4_\sigma| \leq |T^3_\sigma| \Rightarrow \mathcal{F}_{i,m}(LRU \,|\, \sigma) \leq \mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma).$$

Since

$$\mathcal{F}_{i,m}(LRU) = \sum_{\sigma \in \mathcal{I}_{i-1}} \mathcal{F}_{i,m}(LRU \,|\, \sigma)$$

and

$$\mathcal{F}_{i,m}(\mathcal{A}) = \sum_{\sigma \in \mathcal{I}_{i-1}} \mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma),$$

we get $F_{i,m}(LRU) \leq F_{i,m}(\mathcal{A})$. We also have

$$\sum_{\sigma \in \mathcal{I}^f_m} LRU(\sigma) = \sum_{1 \leq i \leq m} \mathcal{F}_{i,m}(LRU)$$

and

$$\sum_{\sigma \in \mathcal{I}^f_m} \mathcal{A}(\sigma) = \sum_{1 \leq i \leq m} \mathcal{F}_{i,m}(\mathcal{A}).$$

Therefore

$$\sum_{\sigma \in \mathcal{I}^f_m} LRU(\sigma) \leq \sum_{\sigma \in \mathcal{I}^f_m} \mathcal{A}(\sigma).$$

Thus LRU (m, f)-dominates $\mathcal{A}$ for every concave* function f, and every integer $m \geq 1$. Hence LRU dominates $\mathcal{A}$. □

COROLLARY 5.1. *For any concave* function f and any paging algorithm $\mathcal{A}$, LRU $\preceq^f_a$ $\mathcal{A}$.*

Therefore LRU is *an* optimal algorithm when we restrict input to sequences with high locality of reference. A natural question is whether or not LRU is a unique optimum, i.e., is there a paging algorithm $\mathcal{A}$ that dominates *LRU*? The following theorem answers this question in the affirmative.

THEOREM 5.1. *No paging algorithm (other than LRU) dominates LRU.*

Proof. Assume for the sake of contradiction that a paging algorithm $\mathcal{A}$ dominates *LRU* and $\mathcal{A}$ is different from *LRU*. Then there must exist a sequence σ so that LRU and $\mathcal{A}$ have different pages in cache after serving σ. Let R denote the set of k distinct pages that are most recently accessed in σ. Let σ' be the smallest suffix of σ that contains k distinct pages, namely pages in R. Select a concave* function f so that σ is consistent with f and $f(|\sigma'|+1) = f(|\sigma'|) = k$.[2]

Consider an arbitrary integer $m > |\sigma|$. We define T_σ, $F_{i,m}(\sigma)$, and T^q_σ for $1 \leq q \leq 4$ as in the proof of Lemma 5.2. Suppose that after serving σ, LRU's cache is different from $\mathcal{A}$'s cache in $1 \leq w \leq k$ pages. Let W be the set of pages that are in $\mathcal{A}$'s cache, but not in LRU's cache. Adding any member of W to σ leads to a sequence that has more than $k = f(|\sigma'|+1)$ distinct pages in a window of size $|\sigma'|+1$; therefore such sequences are not consistent with f and we have $T^4_\sigma = \emptyset$. We have $|T^3_\sigma| \geq w \geq 1$, because for each page $p \in R \setminus W$, the sequence obtained by appending $m - |\sigma|$ repetitions of p to I is consistent with f and belongs to T^3_σ. Therefore $|T^4_\sigma| < |T^3_\sigma|$ and thus $F_{i,m}(LRU \,|\, \sigma) < F_{i,m}(\mathcal{A} \,|\, \sigma)$. Since for any other sequence $\theta \in \mathcal{I}_{i-1}$ we have $F_{i,m}(LRU \,|\, \theta) \leq F_{i,m}(\mathcal{A} \,|\, \theta)$, we get $F_{i,m}(LRU) < F_{i,m}(\mathcal{A})$. Using the proof of Lemma 5.2, we have $F_{i,m}(LRU) \leq F_{i,m}(\mathcal{A})$ for any $1 \leq i \leq m$. Therefore $\sum_{\sigma \in \mathcal{I}^f_m} LRU(\sigma) < \sum_{\sigma \in \mathcal{I}^f_m} \mathcal{A}(\sigma)$. As the argument holds for any $m > |\sigma|$, this refutes the assumption that $\mathcal{A}$ dominates *LRU*. □

THEOREM 5.2. *Let $\mathcal{A}$ be a paging algorithm other than LRU. Then there is a concave* function f so that $\mathcal{A} \not\preceq^f_a$ LRU which implies $\mathcal{A} \not\preceq^f_b$ LRU.*

6 Conclusions

In this paper we introduced Bijective Analysis and Average Analysis as two new techniques for comparing the performance of on-line algorithms. These measures compare on-line algorithms over all sequences of the same length, rather than the worst case sequences alone. In this line of research we set as a goal to propose a model that would more accurately reflect the performance of known on-line paging algorithms. After examining a variety of options, we chose the model of Albers et al. as the best starting point for our endeavours. We then proceeded to further refine the measure using Bijective Analysis. It then became apparent that this model, proposed purely from first principles, would naturally reflect the effects of lookahead and provide separation between well known paging algorithms.

[2]Note that this is not possible if the page immediately before σ' in σ is not in R. In this case, modify σ by repeating the first page of σ'. Since we can assume without loss of generality that both algorithms are demand algorithms, this does not change their behaviour. If the prefix of σ before σ' is not consistent with f, we can apply the same repeating trick to make it consistent with f.

In particular, we showed that these models overcome some of the shortcomings of competitive analysis, namely they reflect the influence of lookahead and separate the performance of LRU and FWF. We proved that all lazy algorithms are equivalent according to Bijective Analysis. Since Bijective Analysis appears to be a natural model, this explains why most measures are unable to separate the theoretical performance of lazy algorithms (e.g. LRU and FIFO).

We then combined Average Analysis with concave analysis, a model of paging with locality of reference proposed by Albers et al. [AFG05]. We showed that under this model, LRU is the sole candidate for being the best on-line paging algorithm. More specifically, we proved that when we restrict the input to sequences with high locality of reference, LRU is never outperformed by another paging algorithm according to Average Analysis, while it outperforms any other paging algorithm according to Average Analysis (and thus Bijective Analysis). Since in practice, input sequences are believed to show locality of reference, this justifies theoretically why LRU is believed to have the best practical performance among on-line paging algorithms. Applying these comparison models to other on-line problems remains an open problem.

Acknowledgements We thank Ian Munro for early discussions on alternative measures of performance for on-line algorithms.

References

[AFG05] Susanne Albers, Lene M. Favrholdt, and Oliver Giel. On paging with locality of reference. *Journal of Computer and System Sciences*, 70, 2005.

[Alb97] Susanne Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, July 1997.

[BDB94] Shai Ben-David and Allan Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11:73–91, 1994.

[Bec04] Luca Becchetti. Modeling locality: A probabilistic analysis of LRU and FWF. In *12th Annual European Symposium on Algorithms (ESA '04)*, pages 98–109, 2004.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BF03] Joan Boyar and Lene M. Favrholdt. The relative worst order ratio for on-line algorithms. In *CIAC: Italian Conference on Algorithms and Complexity*, 2003.

[BFL05] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. The Relative Worst Order Ratio Applied to Paging. In *ACM-SIAM SODA '05*, pages 718–727, 2005.

[BIRS95] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50:244–258, 1995.

[BM04] Joan Boyar and Paul Medvedev. The relative worst order ratio applied to seat reservation. In *SWAT: Scandinavian Workshop on Algorithm Theory*, 2004.

[Den68] Peter J. Denning. The working set model for program behaviour. *Communications of the ACM*, 11(5), May 1968.

[DLO05] Reza Dorrigiv and Alejandro López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 36(3):67–81, September 2005.

[Ken96] Claire Kenyon. Best-fit bin-packing with random order. In *ACM-SIAM SODA '96*, pages 359–364, 1996.

[KP00] Elias Koutsoupias and Christos Papadimitriou. Beyond competitive analysis. *SIAM J. Comput.*, 30:300–317, 2000.

[KPR00] Anna R. Karlin, Steven J. Phillips, and Prabhakar Raghavan. Markov paging. *SIAM Journal on Computing*, 30(3):906–922, 2000.

[PS06] Konstantinos Panagiotou and Alexander Souza. On adequate performance measures for paging. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, pages 487–496, 2006.

[SGG02] Abraham Silberschatz, Peter B. Galvin, and Gerg Gagne. *Operating System Concepts*. John Wiley & Sons, 2002.

[ST85] Daniel D. Sleator and Robert E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28:202–208, 1985.

[Tor98] Eric Torng. A unified analysis of paging and caching. *Algorithmica*, 20(2):175–200, 1998.

[You94] Neal E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, June 1994.

[You98] Neal E. Young. Bounding the diffuse adversary. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA '98)*, pages 420–425, 1998.

[You00] Neal E. Young. On-line paging against adversarially biased random inputs. *Journal of Algorithms*, 37(1):218–235, 2000.

[You02] Neal E. Young. On-line file caching. *Algorithmica*, 33(3):371–383, 2002.