Closing the Gap Between Theory and Practice: New Measures for On-Line Algorithm Analysis

Reza Dorrigiv and Alejandro López-Ortiz

Cheriton School of Computer Science,
University of Waterloo,
Waterloo, ON, N2L 3G1, Canada
{rdorrigiv,alopez-o}@uwaterloo.ca

Abstract. We compare the theory and practice of online algorithms, and show that in certain instances there is a large gap between the predictions from theory and observed practice. In particular, the competitive ratio which is the main technique for analysis of online algorithms is known to produce unrealistic measures of performance in certain settings. Motivated by this we examine first the case of paging. We present a study of the reasons behind this apparent failure of the theoretical model. We then show that a new measure derived from first principles and introduced by [Angelopoulos, Dorrigiv and López-Ortiz, SODA 2007] better corresponds to observed practice. Using these ideas, we derive a new framework termed the cooperative ratio that generalizes to all other online analysis settings and illustrate with examples in list update[1].

1 Introduction

Competitive analysis has long been established as the canonical approach for the analysis of on-line algorithms. Informally, an on-line algorithm processes the input in an on-line manner; that is, the input is a sequence of requests that arrive sequentially in time and the algorithm must make irrevocable decisions with only partial or no knowledge about future requests.

The competitive ratio was formally introduced by Sleator and Tarjan [33], and it has served as a practical framework for studying on-line algorithms. An algorithm (assuming a cost-minimization problem) is said to be α-competitive if the cost of serving any specific request sequence never exceeds α times the optimal cost (up to some additive constant) of an *off-line* algorithm which knows the entire sequence. The competitive ratio has been applied to a variety of problems and settings such as on-line paging, list update, geometric searching/motion planning and on-line approximation of NP-complete problems. Indeed the growth and strength of the field of on-line algorithms is due in no small part to the effectiveness of this measure in the course of practical analysis: the measure is

[1] This paper presents the unifying concepts behind a series of papers on on-line algorithm analysis by the authors, which as a whole lead to a new model for on-line algorithm analysis. See [4,5,18,19,20,21] where aspects of this proposal are discussed separately and at length on their own.

S.-i. Nakano and Md.S. Rahman (Eds.): WALCOM 2008, LNCS 4921, pp. 13–24, 2008.

relatively simple to define yet powerful enough to quantify, to a large extent, the performance of an on-line algorithm. Furthermore computing the competitive ratio has proven to be effective—even in cases where the exact shape of the off-line optimum OPT is unknown.

On the other hand, there are known applications in which competitive analysis yields unsatisfactory results. In some cases it results in unrealistically pessimistic measures, in others it fails to distinguish between algorithms that have vastly differing performance under any practical characterization. Most notably, for the case of paging and on-line motion planning algorithms, competitive analysis does not reflect observed practice, as first noted by Sleator and Tarjan in their seminal paper [33]. Such anomalies have led to the introduction of many alternatives to the competitive analysis of on-line algorithms [8,10,11,13,14,16,22,24,25,27,28,35].

In this paper we study the reasons behind this disconnect, using paging as a case study. We show then that a newly introduced measure by Angelopoulos, Dorrigiv and López-Ortiz [5] refines the model and resolves most of these issues for paging. We then generalize the ideas to other settings which leads to a general framework termed the *adaptive/cooperative ratio* for the analysis of on-line algorithms. This model gives promising results when applied to three well known on-line problems, paging, list update and motion planning. The idea is to normalize the performance of an on-line algorithm by a measure other than the performance of the off-line optimal algorithm OPT. We show that in many instances the performance of OPT on a sequence is a coarse approximation of the difficulty or complexity of this input. Using a finer, more natural measure we can separate paging and list update algorithms which were otherwise indistinguishable under the classical model. This creates a performance hierarchy of algorithms which better reflects the intuitive relative strengths between them. Surprisingly, certain randomized algorithms for paging and list update which are superior to the deterministic optimum in the classical model are not so in the cooperative model. This confirms that the ability of the on-line adaptive algorithm to ignore pathological worst cases can lead to algorithms that are more efficient in practice.

2 Definitions

Let $\sigma = (\sigma_1, \sigma_2, \ldots)$ be an input sequence. We denote by $\sigma_{1:j} = (\sigma_1, \sigma_2, \ldots, \sigma_j)$ the prefix subsequence of the first j requests in σ. An on-line algorithm $\mathcal{A}$ for an optimization problem takes as input a sequence $\sigma = (s_1, s_2, \ldots, s_n)$. The algorithm $\mathcal{A}$ processes the request sequence in order, from σ_1 onwards and produces a partial solution with cost $\mathcal{A}(\sigma_{1:j})$ after the arrival of the jth request (for convenience of notation we will denote as $\mathcal{A}(\sigma) = \text{cost}_{\mathcal{A}}(\sigma)$).

In general it is assumed that the length of the sequence is unknown beforehand and hence an on-line algorithm performs the same steps on the common prefix of two otherwise distinct input sequences. More formally, if σ' is a prefix of σ then $\mathcal{A}(\sigma') = \mathcal{A}(\sigma_{1:|\sigma'|})$. In contrast, the off-line optimal algorithm, denoted as

OPT has access to the entire sequence at once and hence does not necessarily meet the prefix condition.

Definition 1. *An on-line algorithm $\mathcal{A}$ is said to have competitive ratio $C(n)$ if, for all input sequences σ we have:* $\mathcal{A}(\sigma) \leq C(|\sigma|) \cdot \mathrm{OPT}(\sigma)$.

Equivalently, using the more conventional ratio notation, we have that an algorithm is $C(n)$–competitive iff

$$C(n) = \max_{|\sigma|=n, n \geq N_o} \left\{ \frac{\mathcal{A}(\sigma)}{\mathrm{OPT}(\sigma)} \right\}.$$

3 Paging

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 ith page request if the page requested is in the cache (known as a *hit*) it is served at no cost; else, in the case of a *fault* the page is served from slow memory at a cost of one unit. In this event the request results in a cache miss and the on-line algorithm must decide irrevocably which page to evict without the benefit of knowing the *sequence* of upcoming page requests. The goal of a paging algorithm is to minimize the number of faults over the entire input sequence, that is, the *cost* of a particular solution.

Three well known paging algorithms are *Least-Recently-Used* (LRU), *First-In-First-Out* (FIFO), and *Flush-When-Full* (FWF). On a fault, if the cache is full, LRU evicts the page that is least recently requested, FIFO evicts the page that is first brought to the cache, and FWF empties the entire cache. All these paging algorithms have competitive ratio k, which is the best among all deterministic on-line paging algorithms [9].

3.1 Theory Versus Practice

As was mentioned earlier, the standard model for paging does not lead to satisfactory conclusions which are replicated in practice. With the goal of closing the gap between theory and practice, we examine the difference in assumptions between the theoretical competitive ratio model and the practical systems research approach to paging. We now discuss in detail the differences which also appear summarized in Table 1.

1. The theoretical model for the study of paging algorithms is the competitive ratio framework, in contrast, the vast majority of systems research on paging uses the fault rate measure, which simply determines the percentage of page requests leading to a page fault. Consider for example a request sequence of 1M pages, such that an on-line algorithm A has 200 page faults while the off-line optimum has twenty faults. This means that A has a competitive

ratio of 10 which is high, while in terms of the fault rate model A has a page fault rate of 0.002% which is very good.

2. In the worst case one can devise highly contrived request sequences with a very high competitive ratio for any paging algorithm. Since these sequences do not occur naturally, measuring the performance of an online algorithm using them does not shed light on the actual relative performance of various algorithms. Practical studies in contrast use an extensive set of real-life request sequences (traces) gathered from diverse set of applications, over which the performance of any online strategy can be measured.
3. Under the competitive ratio all marking algorithms have the same competitive ratio. In other words LRU and FWF are equal under this measure. In contrast, experimental analysis has consistently shown that LRU and/or minor variants thereof are typically the best choices in practice, while FWF is much worse than LRU. The competitive ratio then fails to separate between these algorithms with very different performance in practice.
4. In terms of practice the theoretical model suggests that LRU might be preferable for "practical" heuristic reasons. In actuality, since paging algorithms are executed concurrently with every page access this limits the complexity of any solution, and hence practical heuristic solutions are simplifications and approximations of LRU.
5. 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 stems from this basic observation. The indirect comparison to an off-line optimal can introduce spurious artifacts due to the comparison of two objects of different types, namely an online and an off-line algorithm. As well the off-line optimum benefits from aspects other than the difficulty of the instances, namely it can take advantage of knowledge of the future, so regardless of the difficulty of servicing a request it might do better as a consequence of this[2]. In contrast the fault rate measure uses a direct comparison of the number of faults per access of paging algorithms to determine which one is preferable.
6. Interestingly, even if algorithms are measured using the competitive ratio, in practice the *worst case* request sequence encountered using LRU has competitive ratio 4, and most sequences measure are well below that with competitive ratio between two and four. Contrast this with the predicted competitive ratio of k under the theoretical model.
7. The off-line optimum model implicitly creates an adversarial model in which the paging algorithm must be able to handle all request sequences, including those maliciously designed to foil the paging algorithm. In contrast, in real life, programmers and compilers purposely avoid bad request sequences and

[2] For example consider the decision whether to purchase car insurance or not [8]. If one purchases insurance then the adversary selects the input in which no claim is filed, if alternatively no insurance is bought then the adversary selects the input in which an accident takes place. In real life, however, it is easy to see that the best on-line strategy is to buy insurance so long as it is priced below the expected loss.

try to arrange the data in a way so as to maximize locality of reference in the request sequence (e.g. the I/O model [1], or the cache oblivious model [30]). In game theoretical terms, the theoretical competitive model is a zero sum game in which the adversary benefits from a badly performing paging algorithm, while in practice paging is a positive sum game in which both the user and the paging algorithm can maximize their respective performances by cooperating and coordinating their strategies. Indeed it has been observed that paging algorithms optimize for locality of reference because this was first observed in real life traces, and now compilers optimize code to increase locality of reference because those paging algorithms excel on those sequences.

8. Lastly, we observe that finite lookahead does not help in the theoretical model, as this is a worst case measure (simply repeat each request for as long as the lookahead is) yet in practice instruction schedulers in many cases know the future request sequence for a small finite lookahead and can use this information to improve the fault rate of paging strategies.

Table 1. Contrast of theory versus practice for paging

Theoretical Model	Systems Framework
Competitive ratio framework	Fault rate measure
Worst case analysis	Typical case analysis
Marking algorithms optimal	LRU and variants thereof are best
In practice LRU is best	LRU is impractical
LFD is off-line optimal	No analogous concept
Competitive ratio is k	Comp. ratio over observed sequences is at most 4
User is a malicious adversary	User (compiler/programmer) seeks locality of reference
No benefit from lookahead	Lookahead helps

3.2 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 [18] for a more comprehensive and detailed exposition.

Loose competitiveness, which was first proposed by Young in [35] and later refined in [38], 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 [28] as well as Young [36,37] 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.

The *Max/Max ratio*, introduced by Borodin and Ben-David [8] 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* [11,12,15] combines some of the desirable properties of the Max/Max ratio and the *random order ratio* (introduced in [27] 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. Recently, Panagiotou and Souza proposed a model that explains the good performance of LRU in practice [29]. 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 a to constant competitive ratio for LRU.

There are several models for paging which assume locality of reference Borodin, Raghavan, Irani, and Schieber [10] 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 [26] proposed a model in which the request sequences are distributed according to a Markov chain process. Becchetti [7] refined the diffuse adversary model of Koutsoupias and Papadimitriou by considering only probabilistic distributions in which temporal locality of reference is present. Torng [34] 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 [2] 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 [17] 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 [2] 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.

4 Bijective Analysis and Average Analysis

Bijective Analysis and Average Analysis are two models recently proposed by Angelopoulos, Dorrigiv and López-Ortiz [5] for comparing on-line algorithms. In this section, we first provide the formal definitions of Bijective Analysis and Average Analysis and then apply them to the paging algorithms. These models have certain desired characteristics for comparing online algorithms: they allow for direct comparison of two on-line algorithms without appealing to the concept of the off-line "optimal" cost (see [5] for a more detailed discussion). In addition, these measures do not evaluate the performance of the algorithm on a single "worst-case" request, but instead use the cost that the algorithm incurs on each and all request sequences. Informally, Bijective Analysis aims to pair input sequences for two algorithms $\mathcal{A}$ and $\mathcal{B}$ using a bijection in such a way that the cost of $\mathcal{A}$ on input σ is no more than the cost of $\mathcal{B}$ on the image of σ, for all request sequences σ of the same length. In this case, intuitively, $\mathcal{A}$ is no worse than $\mathcal{B}$. On the other hand, Average Analysis compares the average cost of the two algorithms over all request sequences of the same length. 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.

Definition 2. *[5] 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 $n_0 \geq 1$ so 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 $\mathcal{A} \equiv_b \mathcal{B}$. Lastly 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 $\mathcal{A} \prec_b \mathcal{B}$.*

Definition 3. *[5] 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 $n_0 \geq 1$ so 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.*

In [5] it is shown that LRU is strictly better than FWF under Bijective Analysis. Additionally, lookahead is beneficial in Bijective Analysis model: more specifically, 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. Both of these results describe natural, "to-be-expected" properties of the corresponding paging strategies which competitive analysis nevertheless fails to yield.

Also it turns out that a very large class of natural paging strategies known as *lazy* algorithms (including LRU and FIFO, but not FWF) are in fact strongly equivalent under this rather strict bijective measure. The strong equivalence of lazy algorithms is evidence of an inherent difficulty to separate these algorithms in any general unrestricted 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. [2]) or assume a distribution (or a set of distributions) on the sequence space (e.g. as in Koutsoupias and Papadimitriou [28], Young [36] and Becchetti [7]). The latter group of approaches use probabilistic assumptions on the sequence space. However, we are interested in measures that separate algorithms under a deterministic model.

Next we briefly describe concave analysis. In this model 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 number of distinct pages in a window of size n within a given request sequence. Extensive experiments with real data show that this function can be bounded by a concave function for most practical request sequences [2]. Let f be an increasing concave function. 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 $n \in \mathcal{N}$. Now we can model locality by considering only those request sequences that are consistent with f.

Using a combination of Average Analysis and concave analysis, Angelopoulos et al. [5] show that LRU is never outperformed in any possible subpartition on the request sequence space induced by concave analysis, while it always outperforms *any other paging algorithm* in at least one subpartition of the sequence space. This result proves separation between LRU and all other algorithms and provides theoretical backing to the observation that LRU is preferable in practice.

To be more precise we restrict the input sequences to those consistent with a given concave function f. Let $\mathcal{I}^f$ denote the set of such sequences. We can easily modify the definitions of Bijective Analysis and Average Analysis (Definition 2 and Definition 3) 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. 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. $\mathcal{I}_n^f$.

Theorem 1 (Unique optimality of LRU). *[5] For any concave function f and any paging algorithm $\mathcal{A}$, LRU $\preceq_a^f$ $\mathcal{A}$. Furthermore, let $\mathcal{A}$ be a paging algorithm other than LRU. Then there is a concave function f so that $\mathcal{A} \not\preceq_a^f$ LRU which implies $\mathcal{A} \not\preceq_b^f$ LRU.*

5 List Update and Cooperative Analysis

List update is a fundamental problem in the context of on-line computation. Consider an unsorted list of l items. The input to the algorithm is a sequence of n requests that should be served in an on-line manner. Let $\mathcal{A}$ be an arbitrary

on-line list update algorithm. To serve a request to an item x, $\mathcal{A}$ should linearly search the list until it finds x. If x is the ith item in the list, $\mathcal{A}$ incurs cost i to access x. Immediately after accessing x, $\mathcal{A}$ can move x to any position closer to the front of the list at no extra cost. This is called a *free exchange*. Also $\mathcal{A}$ can exchange any two consecutive items at a cost of 1. These are called *paid exchanges*. An efficient algorithm should use free and paid exchanges so as to minimize the overall cost of serving a sequence. This is called the *standard cost model* [3]. Three well-known deterministic on-line algorithms are *Move-To-Front* (MTF), *Transpose*, and *Frequency-Count* (FC). MTF moves the requested item to the front of the list whereas Transpose exchanges the requested item with the item that immediately precedes it. FC maintains a frequency count for each item, updates this count after each access, and makes necessary moves so that the list always contains items in non-increasing order of frequency count. Sleator and Tarjan showed that MTF is 2-competitive, while Transpose and FC do not have constant competitive ratios [33].

The competitive analysis of list update algorithms does not have as many drawbacks as paging and at first it gives promising results: list update algorithms with better competitive ratio tend to have better performance in practice. However, in terms of separation list update algorithms have similar drawbacks to paging: while algorithms can generally be more easily distinguished than in the paging case, the experimental study of list update algorithms by Bachrach and El-Yaniv suggests that the relative performance hierarchy as computed by the competitive ratio does not correspond to the observed relative performance of the algorithms in practice [6].

Like paging, "real-life" input sequences for list update problem usually exhibit *locality of reference*. As stated before, for the paging problem, several models for capturing locality of reference have been proposed [2,7,34]. Likewise, many researchers have pointed out that input sequences of list update algorithms in practice show locality of reference [9,23,32] and actually on-line list update algorithms try to take advantage of this property [23,31]. Hester and Hirschberg [23] posed the question of providing a good definition of locality of accesses for the list update problem as an open problem. In addition, it has been commonly assumed, based on intuition and experimental evidence, that MTF is the best algorithm on sequences with high locality of reference, e.g., Hester and Hirschberg [23] claim: "move-to-front performs best when the list has a high degree of locality". However, to the best of our knowledge, locality of reference for list update algorithms had not been formally studied, until recently [4,19].

In [4], Angelopoulos, Dorrigiv and López-Ortiz extended the concave analysis model [2] to the list update problem. The validity of the extended model was supported by experimental results obtained on the Calgary Corpus, which is frequently used as a standard benchmark for evaluating the performance of compression algorithms (and by extension list update algorithms, e.g. [6]). They combined Average Analysis with concave analysis and proved that under this model MTF is never outperformed, while it always outperforms *any other*

on-line list update algorithm. Thus, [4] resolved the open problem posed by Hester and Hirschberg [23].

Based on adaptive analysis ideas, Dorrigiv and López-Ortiz [19] proposed cooperative analysis for analyzing on-line algorithms. The idea behind *cooperative analysis* is to give more weight to "well-behaved" input sequences. Informally, an on-line algorithm has good *cooperative* ratio if it performs well on good sequences and not too poorly on bad sequences. For example, as stated before, input sequences for paging and list update have *locality of reference* in practice, therefore one possibility is to relate goodness of sequences to their amount of locality. In [19], we showed that cooperative analysis of paging and list update algorithms gives promising results. Here we just briefly describe the results for list update. We use a measure of badness that is related to locality of reference as follows. For a sequence σ of length n, define $d_\sigma[i]$ for $1 \leq i \leq n$ as either 0 if this is the first request to item $\sigma[i]$, or otherwise, the number of distinct items that are requested since the last request to $\sigma[i]$ (including $\sigma[i]$). Define $\overline{\ell(\sigma)}$, the non-locality of a sequences σ, as $\overline{\ell(\sigma)} = \sum_{1 \leq i \leq n} d_\sigma[i]$.

Definition 4. *[19] We say that an on-line list update algorithm $\mathcal{A}$ has locality-cooperative ratio α if there is a constant β so that for every sequence σ, $\mathcal{A}(\sigma) \leq \alpha \times \overline{\ell(\sigma)} + \beta$. We define locality-cooperative ratio of $\mathcal{A}$, $LCR(\mathcal{A})$, as the smallest number α so that $\mathcal{A}$ has locality-cooperative ratio α.*

The following theorem summarizes the results proved for locality-cooperative ratio of list update algorithms.

Theorem 2. *[19] For any on-line list update algorithm $\mathcal{A}$, $1 \leq LCR(\mathcal{A}) \leq l$; furthermore:*

1. $LCR(MTF) = 1$.
2. $LCR(Transpose) \geq l/2$.
3. $LCR(FC) \geq \frac{l+1}{2} \approx l/2$.
4. $LCR(TS) \geq \frac{2l}{l+1} \approx 2$.
5. $LCR(Bit) \geq \frac{3l+1}{2l+2} \approx 3/2$.

6 Conclusions

In this paper, we highlighted the gap between theoretical and experimental results for some on-line problems and possible ways to close this gap. We observed that standard measure for analysis of on-line algorithms, i.e., competitive analysis, leads to results that are not consistent with practice for paging and list update. Then we described reasons for the shortcomings of competitive analysis and described several new models for analysis of on-line algorithms that do not have these drawbacks. Bijective Analysis and Average Analysis directly compare two on-line algorithms on all sequences of the same length and lead to satisfactory results when applied to paging and list update. The new concept of cooperative ratio applies adaptive analysis ideas to the analysis of on-line algorithms and divides the cost of the algorithm on a sequence to some property of that sequence.

References

1. Aggarwal, A., Vitter, J.S.: The Input/Output complexity of sorting and related problems. Communications of the ACM 31(9), 1116–1127 (1988)
2. Albers, S., Favrholdt, L.M., Giel, O.: On paging with locality of reference. Journal of Computer and System Sciences 70(2), 145–175 (2005)
3. Albers, S., Westbrook, J.: Self-organizing data structures. In: Fiat, A. (ed.) Online Algorithms. LNCS, vol. 1442, pp. 13–51. Springer, Heidelberg (1998)
4. Angelopoulos, S., Dorrigiv, R., López-Ortiz, A.: List update with locality of reference: Mtf outperforms all other algorithms. Technical Report CS-2006-46, University of Waterloo, Cheriton School of Computer science (November 2006)
5. Angelopoulos, S., Dorrigiv, R., López-Ortiz, A.: On the separation and equivalence of paging strategies. In: SODA 2007. Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms, pp. 229–237 (2007)
6. Bachrach, R., El-Yaniv, R.: Online list accessing algorithms and their applications: Recent empirical evidence. In: SODA 1997. Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 53–62 (1997)
7. Becchetti, L.: Modeling locality: A probabilistic analysis of LRU and FWF. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, pp. 98–109. Springer, Heidelberg (2004)
8. Ben-David, S., Borodin, A.: A new measure for the study of on-line algorithms. Algorithmica 11, 73–91 (1994)
9. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
10. Borodin, A., Irani, S., Raghavan, P., Schieber, B.: Competitive paging with locality of reference. Journal of Computer and System Sciences 50, 244–258 (1995)
11. Boyar, J., Favrholdt, L.M.: The relative worst order ratio for on-line algorithms. In: Proceedings of the 5th Italian Conference on Algorithms and Complexity (2003)
12. Boyar, J., Favrholdt, L.M., Larsen, K.S.: The relative worst order ratio applied to paging. In: SODA 2005. Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms, pp. 718–727 (2005)
13. Boyar, J., Larsen, K.S.: The Seat Reservation Problem. Algorithmica 25(4), 403–417 (1999)
14. Boyar, J., Larsen, K.S., Nielsen, M.N.: The Accommodating Function: A generalization of the competitive ratio. SIAM Journal on Computing 31(1), 233–258 (2001)
15. Boyar, J., Medvedev, P.: The relative worst order ratio applied to seat reservation. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 90–101. Springer, Heidelberg (2004)
16. Chrobak, M., Noga, J.: LRU is better than FIFO. Algorithmica 23(2), 180–185 (1999)
17. Denning, P.J.: The working set model for program behaviour. Communications of the ACM, 11(5) (May 1968)
18. Dorrigiv, R., López-Ortiz, A.: A survey of performance measures for on-line algorithms. SIGACT News (ACM Special Interest Group on Automata and Computability Theory) 36(3), 67–81 (2005)
19. Dorrigiv, R., López-Ortiz, A.: The cooperative ratio of on-line algorithms. Technical Report CS-2007-39, University of Waterloo, Cheriton School of Computer science (October 2007)

20. Dorrigiv, R., López-Ortiz, A.: On certain new models for paging with locality of reference. In: WALCOM 2008. Proceedings of the 2nd Workshop on Algorithms and Computation (to appear, 2008)
21. Dorrigiv, R., López-Ortiz, A., Munro, J.I.: On the relative dominance of paging algorithms. In: ISAAC 2007. Proceedings of the 18th International Symposium on Algorithms and Computation (to appear, 2007)
22. Fiat, A., Woeginger, G.J.: Competitive odds and ends. In: Fiat, A. (ed.) Online Algorithms. LNCS, vol. 1442, pp. 385–394. Springer, Heidelberg (1998)
23. Hester, J.H., Hirschberg, D.S.: Self-organizing linear search. ACM Computing Surveys 17(3), 295 (1985)
24. Irani, S.: Competitive analysis of paging. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms. LNCS, vol. 1442, pp. 52–73. Springer, Heidelberg (1998)
25. Irani, S., Karlin, A.R., Phillips, S.: Strongly competitive algorithms for paging with locality of reference. SIAM Journal on Computing 25, 477–497 (1996)
26. Karlin, A.R., Phillips, S.J., Raghavan, P.: Markov paging. SIAM Journal on Computing 30(3), 906–922 (2000)
27. Kenyon, C.: Best-fit bin-packing with random order. In: SODA 1996. Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 359–364 (1996)
28. Koutsoupias, E., Papadimitriou, C.: Beyond competitive analysis. SIAM Journal on Computing 30, 300–317 (2000)
29. Panagiotou, K., Souza, A.: On adequate performance measures for paging. In: STOC 2006. Proceedings of the 38th Annual ACM Symposium on Theory of Computing, pp. 487–496 (2006)
30. Prokop, H.: Cache-oblivious algorithms. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science (1999)
31. Reingold, N., Westbrook, J., Sleator, D.D.: Randomized competitive algorithms for the list update problem. Algorithmica 11, 15–32 (1994)
32. Schulz, F.: Two new families of list update algorithms. In: Chwa, K.-Y., Ibarra, O.H. (eds.) ISAAC 1998. LNCS, vol. 1533, pp. 99–108. Springer, Heidelberg (1998)
33. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM 28, 202–208 (1985)
34. Torng, E.: A unified analysis of paging and caching. Algorithmica 20(2), 175–200 (1998)
35. Young, N.E.: The k-server dual and loose competitiveness for paging. Algorithmica 11(6), 525–541 (1994)
36. Young, N.E.: Bounding the diffuse adversary. In: SODA 1998. Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 420–425 (1998)
37. Young, N.E.: On-line paging against adversarially biased random inputs. Journal of Algorithms 37(1), 218–235 (2000)
38. Young, N.E.: On-line file caching. Algorithmica 33(3), 371–383 (2002)