On Developing New Models, with Paging as a Case Study*

Reza Dorrigiv Alejandro López-Ortiz †
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ont., N2L 3G1, Canada
{rdorrigiv,alopez-o}@uwaterloo.ca

Abstract

As computer science has progressed, numerous models and measures have been developed over the years. Among the most commonly used in theoretical computer science are the RAM model, the I/O model, worst case analysis, space (memory) usage, average case analysis, amortized analysis, adaptive analysis and the competitive ratio. New models are added to this list every few years to reflect varying constraints imposed by novel application or advances in computer architectures. Examples of alternative models are the transdichotomous RAM or word-RAM, the data stream model, the MapReduce model, the cache oblivious model and the smoothed analysis model. New models and measures, when successful expand our understanding of computation and open new avenues of inquiry. As it is to be expected relatively few models and paradigms are introduced every year, and even less are eventually proven successful.

In this paper we discuss first certain shortcomings of the online competitive analysis model particularly as it concerns paging, discuss existing solutions in the literature as well as present recent progress in developing models and measures that better reflect actual practice for the case of paging. From there we proceed to a more general discussion on how to measure and evaluate new models within theoretical computer science and how to contrast them, when appropriate, to existing models. Lastly, we highlight certain "natural" choices and assumptions of the standard worst-case model which are often unstated and rarely explicitly justified. We contrast these choices to those made in the formalization of probability theory.

1 Introduction

Starting with the seminal paper of Sleator and Tarjan in 1985 it was observed that the competitive ratio did not always fairly evaluate online algorithms. Since then many alternative measures have been proposed to remedy this fact. Yet, a quick perusal of the systems literature as it applies to paging suggests that none of the many models suggested by theory have consistently been put to good use in practice (see Section 2 as well as our previous survey [DLO05]).

In this paper we continue work described in our previous survey. First we discuss the key differences between how paging algorithms are evaluated in theory and in practice (Section 2), we then survey the search for alternatives to the competitive ratio and highlight new developments (Section 3). In Section 4 we discuss a new and effective measure that resolves most of the issues

*Portions of this paper appeared in preliminary form in WALCOM'08 and WAOA'09.

with paging algorithms. This naturally leads to a discussion of the features of the standard worst-case model. We observe that some of the choices seem natural mostly out of tradition rather than fundamental reasons. As a point of comparison of alternative approaches we contrast this with the formalization of probability theory, in which many of these choices are left open. Evidence suggests that perhaps the probability theory model while very general, is unnecessarily so. In turn, the worst case model has a set of ready made choices which have proven rather effective, but are not unique or always optimal in all settings (Section 5). We conclude with a discussion of the characteristics of successful models and measures (Section 6).

We consider first the case of online algorithms, for which there is a well accepted theoretical model. As we shall see, in certain circumstances this model has been found wanting by practitioners. This has been observed by numerous other researchers, and hence there is a suite of existing alternative models.

2 Online Algorithms and Competitive Analysis

Competitive analysis has long been established as the canonical approach for the analysis of online algorithms. Informally, the input of an online algorithm 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—formally introduced by Sleator and Tarjan [ST85a]—has served as a practical framework for the study of online algorithms. An algorithm (assuming a cost-minimization problem) is said to be c-competitive if the cost of serving any specific request sequence never exceeds c times the optimal cost (up to some additive constant) of an *offline* algorithm which knows the entire sequence. The competitive ratio has been applied to a variety of problems and settings such as classical online algorithms, geometric searching, motion planning and online approximation of NP-complete problems. Indeed the growth and strength of the field of online algorithms is due in no small part to the effectiveness of this measure in the course of practical analysis: the measure is relatively simple to define yet powerful enough to quantify, to a large extent, the performance of an online algorithm. Furthermore computing the competitive ratio has proven to be effective—even in cases where the exact shape of the offline 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 online motion planning algorithms, competitive analysis does not reflect observed practice, as first noted by Sleator and Tarjan in their seminal paper [ST85a]. Such anomalies have led to the introduction of many alternatives to the competitive analysis of online algorithms [BDB94, BL99, BLN01, BF03, KP00, Ken96, FW98, Ira98, BIRS95, IKP96, CN99, You94].

In this section we study the reasons behind the gap between theory and practice for online algorithm analysis . We then highlight attempts—by the authors as well as others—to resolve this disconnect by means of improving and refining the model. We discuss the benefits and drawbacks for each of these proposed measures.

2.1 Paging: A Case Study

Paging is a fundamental problem in the context of the analysis of online 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 online 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 online paging algorithms [BEY98].

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 online algorithm $\mathcal{A}$ for an optimization problem takes as input a sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$. The algorithm $\mathcal{A}$ processes the request sequence in order, from σ_1 onwards and produces a partial solution with cost $cost_{\mathcal{A}}(\sigma_{1:j})$ after the arrival of the jth request.

In general it is assumed that the length of the sequence is unknown beforehand and hence an online 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 $cost_{\mathcal{A}}(\sigma') = cost_{\mathcal{A}}(\sigma_{1:|\sigma'|})$. In contrast, the offline optimal algorithm, denoted as OPT has access to the entire sequence at once and hence does not necessarily meet the prefix condition. Equivalently, using the more conventional ratio notation, we have that an algorithm is $C(n)$–competitive iff

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

2.2 The Gap between Theory and 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 the 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. *Competitive ratio v. Fault rate.* The theoretical model for the study of paging algorithms is the competitive ratio framework in which the preformance is measured relative to an idealized offline optimal algorithm. In contrast, the vast majority of systems research on paging uses the fault rate measure, which simply determines the percentage of page requests in the input sequence leading to a page fault. Consider for example a request sequence of 1M pages, such that an online algorithm $\mathcal{A}$ has 200 page faults while the offline optimum has twenty faults. This means that $\mathcal{A}$ has a competitive ratio of 10 which is high, while in terms of the fault rate model $\mathcal{A}$ has a page fault rate of 0.002% which is very good.

2. *Worst case v. Typical case.* 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 on 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. *Marking algorithms v.* LRU. Under the competitive analysis all marking algorithms have the same performance. In other words LRU and FWF are equal under the competitive ratio 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 these algorithms with very different performance in practice.

4. *Practical algorihtms.* 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. *Idealized offline optimum.* Competitive analysis uses an optimal offline algorithm as a baseline to compare online algorithms. While this may be convenient, it is rather indirect: one could argue that in comparing $\mathcal{A}$ to $\mathcal{B}$ all we need to study is the relative cost of the algorithms on the request sequences. The indirect comparison to an offline optimal can introduce spurious artifacts due to the comparison of two objects of different types, namely an online and an offline algorithm. As well the offline 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[1]. 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. *Unlimited v. Restricted input space.* 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 result in a competitive ratio well below that. Contrast this with the predicted competitive ratio of k under the theoretical model.

7. *Adversarial v. Cooperative model.* The offline 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 practice, programmers and compilers purposely avoid bad request sequences and 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 [AV88], or the cache oblivious model [Pro99]). In game theoretical terms, the 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

[1]For example consider the decision whether to purchase car insurance or not [BDB94]. 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 online strategy is to buy insurance so long as it is priced below the expected loss.

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 offline optimal	No analogous concept
Competitive ratio is k	Observed Comp. ratio is at most 4
User is a malicious adversary	User seeks locality of reference
No benefit from lookahead	Lookahead helps

Table 1: Contrast of theory versus practice for paging.

the paging algorithm can maximize their respective performances by cooperating and coordinating their strategies. Indeed it was observed by Denning that paging algorithms optimize for locality of reference because this was once observed in real life traces, and now compilers optimize code to increase locality of reference because now paging algorithms excel on those sequences. In other words, locality of reference which was once observed "in the wild" is now actively sought after by both programmers and paging algorithms and hence its existence is reinforced.

8. *Effect of lookahead.* 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 use this information to improve the fault rate of paging strategies in practice.

3 The Search for Alternatives

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 offline 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 online 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.

The *Max/Max ratio*, introduced by Borodin and Ben-David [BDB94] compares online algorithms based on their amortized worst-case behaviour where 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 online bin packing problem. As with the Max/Max ratio, it allows for direct comparison of two online 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 [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 a to constant competitive ratio for LRU.

Bijective analysis and Average analysis recently proposed by Angelopoulos, Dorrigiv and López-Ortiz [ADLO07], build upon the framework of [AFG05]. These two models have certain desirable characteristics for comparing online algorithms: they allow for direct comparison of two online algorithms without appealing to the concept of the offline "optimal" cost (see [ADLO07] 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 online 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.

We say that an online algorithm $\mathcal{A}$ is *no worse* than an online 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 $\mathcal{B} \preceq_b \mathcal{A}$. 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 $\mathcal{A} \prec_b \mathcal{B}$ [ADLO07].

We say that an online algorithm $\mathcal{A}$ is *no worse* than an online 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 [ADLO07].

As it turns out, 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. [AFG05]) or assume a distribution (or a set of distributions) 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, we are interested in measures that separate algorithms under a deterministic model.

It is well known that input sequences for paging and several other problems show locality of reference. This means that when a page is requested it is more likely to be requested in the near future. Therefore several models for paging with locality of reference have been proposed. In the early days of computing, Denning recognized the locality of reference principle and modeled it using

the well known *working set* model [Den68, Den80]. He defined the working set of a process as the set of most recently used pages and addressed thrashing using this model. According to [Den05], the word "locality" was first used by Denning in discussions with Dennis and Belady. They noticed that programs showed locality behavior even when it was not explicitly planned. After the introduction of the working set model, the locality principle has been adopted in operating systems, databases, hardware architectures, compilers, and many other areas. Therefore it holds even more so today. Indeed, [Den05] states "locality of reference is one of the cornerstones of computer science."

One apparent reason for the drawbacks of competitive analysis of paging is that it does not incorporate the corresponding locality of reference. There are several models for paging which assume 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 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.

Hence the need to combine Bijective analysis with an assumption of locality of reference model such as 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 $D()$ that represents the maximum number of distinct pages in a window of size n within a given request sequence. This gives a non-decreasing function which, furthermore, is trivially bounded by a concave function, namely the identity. Extensive experiments with real data show that in fact this function can generally be bounded by a non-trivial concave function $f()$ which approximates $D()$ rather well for most practical request sequences [AFG05]. 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 $n \in \mathbb{N}$. Now we can model locality by considering only those request sequences that are consistent with f.

Using this measure Angelopoulos et al. [ADLO07] 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. This is the first deterministic theoretical model to provide full separation between LRU and all other algorithms.

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 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**)** *[ADLO07] 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.

Recently this result was strengthened by Angelopolous and Schweitzer [AS09] where they showed that the separation also holds under the stricter Bijective analysis (as opposed to Average analysis) using the concave analysis framework. However Bijective analysis and Average analysis are not particularly effective measures, hence even though separation for LRU was finally achieved, the quest for a better model remains. In the next Section we will discuss a new, more effective model for the analysis of online algorithms.

4 Adaptive Analysis

Standard algorithm analysis expresses performance in terms of the input size. Adaptive analysis takes into account the difficulty of input instances as well. This means that an algorithm has good performance according to adaptive analysis if it performs well on "easy" instances and not too poorly on "difficult" ones. We define adaptive performance of an algorithm by representing its traditional performance in terms of both the input size and the difficulty of the input. In general, adaptive analysis involves two key steps: (i) find a realistic difficulty measure for input instances and (ii) propose algorithms that provably perform well under such a measure. Observe that the competitive ratio can be seen as a special case of adaptive analysis, namely the case where the measure of difficulty is the performance of the offline OPT.

In terms of (i), namely identifying a reasonable measure of difficulty, this is generally a straightforward task. In most cases it is easy to identify a reasonable measures of difficulty. Indeed the challenge is more often than not to justify a specific choice among several natural ones. Empirically it has been observed that for most problems the differences in difficulty measure do not radically alter the nature of the optimal algorithms for those measures. Observe that this situation is analogous to the encoding of the input under the classical worst-case model: in general there are many reasonable ways to encode the input, however under linear time online decoding schemes most algorithms remain optimal by adding a simple preprocessing step in the beginning.

Adaptive analysis brings to online algorithms the ability to use a finer measure of difficulty. For each problem, we can choose the measure that best reflects the difficulty of the input. It is unlikely that the offline OPT is such a measure for all cases. Another feature of adaptive analysis of online algorithms is that in certain online problems, the competitive ratio measure might force the algorithm to make a move that is suboptimal in most cases except for a pathological worst case scenario. If the application is such that these pathological cases are agreed to be

of lesser importance, then the online strategy can perform somewhat more poorly in these and make the choice that is best for the common case. Observe that the input is no longer assumed to be adversarially constructed. This better reflects, in particular, the case of paging, in which programmers, compilers and optimized virtual machines (such as JVMs) go to great lengths to maintain and increase locality of reference in the code. Hence it is more realistic to assume that paging sequences are not adversarial and that furthermore, the user/programmer fully expects code with low locality of reference to show a degradation in performance. The same observation has been made in scenarios such as online robot exploration [LOS96, IKM93, HIKK01, LO96, LOS04, LOS01a] and network packet switching, in which a robot vacuuming a room or a router serving a packet sequence need only concentrate on well-behaved cases. A vacuuming robot need not efficiently vacuum a maze, neither does the router have to keep up with denial-of-service floods.

Dorrigiv and López-Ortiz [DLO07] initially proposed cooperative analysis as a new framework for the analysis of online algorithms and applied it to paging and list update problems. This measure was further refined by Dorrigiv et al. in [DELO09] where it is reframed in terms of adaptive and/or parameterized analysis.

In this section we apply parameterized analysis to paging. Paging and list update are ideal testbeds for developing alternative measures, given our extensive understanding of these problems. We know why and where competitive analysis fails, what are typical sequences in practice and how to evaluate a new technique to determine if it indeed overcomes the known shortcomings. It is important to note that even though well studied, most of the alternative models for these problems are only partially successful in resolving the issues posed by these testbeds and as such these problems are still challenging case studies against which to test a new model. We express the performance of well known paging algorithms in terms of a measure of locality of reference. As we shall see, for paging, this leads to better separation than the competitive ratio. Furthermore, in contrast to competitive analysis it reflects the intuition that a larger cache leads to a better performance. We also provide experimental results that justify the applicability of our measure in practice. Some of the proofs follow the general outline of standard competitive analysis proofs (e.g., those in [BEY98]), which shows the power of the adaptive model.

As stated before, several studies have been presented that integrate the concept of locality for paging, culminating with the LRU separation results in [ADLO07, AS09]. These separation results are based on heavy machinery specifically designed to resolve this singular long-standing question. In contrast, the new measure we propose is easier to apply and creates a performance hierarchy of paging and list update algorithms which better reflects their intuitive relative strengths. Several previously observed experimental properties can be readily proven using the new model. This is a strength of the new model in that is effective, that is readily applicable to a variety of algorithms and with meaningful results.

As stated before input sequences for paging show locality of reference in practice. We want to express the performance of paging algorithms on a sequence in terms of the amount of the locality in that sequence. Therefore we need a measure that assigns a number proportional to the amount of locality in each sequence. None of the previously described models provide a unique numerical value as a measure of locality of reference. We define a quantitative measure for non-locality of paging instances.

Definition 1 *For a sequence σ we define $d_\sigma[i]$ as either $k+1$ if this is the first request to page $\sigma[i]$, or otherwise, the number of distinct pages that are requested since the last request to $\sigma[i]$ (including*

	espresso	li	eqntott	compress	tomcatv	ear	sc	swm	gcc
Distinct	3913	3524	9	189	5260	1614	561	3635	2663
$\overline{\lambda}$	193.1	195.2	1.7	2.3	348.3	34.1	5.4	166.7	90.6
Ratio	4.9%	5.5%	19.3%	1.2%	6.6%	2.1%	1.0%	4.6%	3.4%

Table 2: Locality of address traces collected from SPARC processors running the SPEC92 benchmarks.

$\sigma[i]$).[2] *Now we define* $\overline{\lambda}(\sigma)$*, the "non-locality" of* σ*, as* $\overline{\lambda}(\sigma) = \frac{1}{|\sigma|}\sum_{1\le i\le|\sigma|} d_\sigma[i]$*. We denote the non-locality by* $\overline{\lambda}$ *if the choice of* σ *is clear from the context.*

If σ has high locality of reference, the number $d_\sigma[i]$ of distinct pages between two consecutive requests to a page is small for most values of i and thus σ has a low non-locality. Note that while this measure is related to the working set model [Den68] and the locality model of [AFG05], it differs from both in several aspects. Albers et al. [AFG05] consider the maximum/average number of distinct pages in all windows of the same size, while we consider the number of distinct pages requested since the last access to each page. Also our analysis does not depend on a concave function f whose identification for a particular application might not be straightforward. Our measure is also closely related to the working set theorem in area of self-organizing data structures [ST85b]. For binary search trees (like splay trees), the working set bound is defined as $\sum_{1\le i\le|\sigma|}\log(d_\sigma[i]+1)$. The logarithm can be explained by the logarithmic bounds on most operations in binary search trees. Thus our measure of locality of reference can be considered as variant of this measure in which we remove the logarithm.

Experimental Evaluation of the Measure. In order to check validity of our measure we ran some experiments on traces of memory reference streams from the NMSU TraceBase [Uni]. Here we present the results of our experiments on address traces collected from SPARC processors running the SPEC92 benchmarks. We considered a page size of 2048 bytes and truncated them after 40000 references. The important thing to notice is that these are not special cases or artificially generated memory references, but are access patterns a real-life implementation of any paging algorithm might face. The results for the corresponding eleven program traces are shown in Table 2. The first row shows the number of distinct pages, the second row shows $\overline{\lambda}$, and finally the third row shows the ratio of the actual locality to the worst possible locality. The worst possible locality of a trace asymptotically equals the number of distinct pages in that trace. It is clear from the low ratios that in general these traces exhibit high locality of reference as defined by our measure.

Next we analyze several well known paging algorithms in terms of the non-locality parameter. We consider the fault rate, the measure usually used by practitioners. The fault rate of a paging algorithm $\mathcal{A}$ on a sequence σ is defined as $\mathcal{A}(\sigma)/|\sigma|$, i.e., the number of faults $\mathcal{A}$ incurs on σ normalized by the length of σ. The fault rate of $\mathcal{A}$, $FR(A)$, is defines as the asymptotic worst case fault rate of $\mathcal{A}$ on any sequence. The bounds are in the worst case sense, i.e., when we say $FR(\mathcal{A}) \ge f(\overline{\lambda})$ we mean that there is a sequence σ such that $\frac{\mathcal{A}(\sigma)}{|\sigma|} \ge f(\overline{\lambda}(\sigma))$ and when we say $FR(\mathcal{A}) \le g(\overline{\lambda})$ we mean that for every sequence σ we have $\frac{\mathcal{A}(\sigma)}{|\sigma|} \le g(\overline{\lambda}(\sigma))$. Also for simplicity, we ignore the details related to the special case of the first few requests (the first block or phase).

[2] Asymptotically, and assuming the number of requests is much larger than the number of distinct pages, any constant can replace $k+1$ for the $d_\sigma[i]$ of the first accesses.

Algorithm	Lower Bound	Upper Bound
Deterministic	$\frac{\overline{\lambda}}{k+1}$	$\frac{\overline{\lambda}}{2}$
LRU	$\frac{\overline{\lambda}}{k+1}$	$\frac{\overline{\lambda}}{k+1}$
Marking	$\frac{\overline{\lambda}}{k+1}$	$\frac{2\overline{\lambda}}{k+3}$
FWF	$\frac{2\overline{\lambda}}{k+3}$	$\frac{2\overline{\lambda}}{k+3}$
FIFO	$\frac{\overline{\lambda}}{k+1}$	$\frac{\overline{\lambda}}{k+1}$
LFU	$\frac{2\overline{\lambda}}{k+3}$	$\frac{\overline{\lambda}}{2}$
LIFO	$\frac{\overline{\lambda}}{2}$	$\frac{\overline{\lambda}}{2}$
LRU-2	$\frac{2k\overline{\lambda}}{(k+1)(k+2)}$	$\frac{2\overline{\lambda}}{k+1}$
MARK	$\frac{2\overline{\lambda}}{3k+1}$	$\frac{2\overline{\lambda}}{3k+1}$
LFD	$\frac{\overline{\lambda}}{3k+1}$	$\frac{2\overline{\lambda}}{3k+1}$

Table 3: Bounds for paging.

Asymptotically and as the size of the sequences grow, this can only change the computation by additive lower order terms.

We give a proof of the first two results to illstrate the effectiveness of the model and refer the reader to [DELO09] for a full proof of the results summarized in Table 3.

Lemma 1 *For any deterministic paging algorithm $\mathcal{A}$, $\frac{\overline{\lambda}}{k+1} \leq FR(\mathcal{A}) \leq \frac{\overline{\lambda}}{2}$.*

Proof: For the lower bound consider a slow memory containing $k+1$ pages. Let σ be a sequence of length n obtained by first requesting $p_1, p_2, \ldots, p_k, p_{k+1}$, and afterwards repeatedly requesting the page not currently in $\mathcal{A}$'s cache. Since $\frac{\mathcal{A}(\sigma)}{|\sigma|} = n/n = 1$, and $\overline{\lambda}$ is at most $k+1$ (there are $k+1$ distinct pages in σ), the lower bound follows.

For the upper bound, consider any request sequence σ of length n. If the i^{th} request is a fault charged to $\mathcal{A}$, then $d_\sigma[i] \geq 2$ (otherwise $\sigma[i]$ cannot have been evicted). Hence, $2\mathcal{A}(\sigma) \leq \sum_{i=1}^{n} d_\sigma[i]$ and the upper bound follows. □

We now show that LRU attains the best possible performance in terms of $\overline{\lambda}$.

Theorem 2 $FR(\text{LRU}) = \frac{\overline{\lambda}}{k+1}$.

Proof: It follows from the observation that LRU faults on the request $\sigma[i]$ if and only if $d_\sigma[i] \geq k+1$, which implies $\text{LRU}(\sigma) \leq \frac{\overline{\lambda}}{k+1}$ and Lemma 1. □

According to the results of Table 3, LRU and FIFO have optimal performance among deterministic algorithms. Marking algorithms can be twice as bad and FWF is among the worst marking algorithms. LIFO has the worst performance possible and LRU-2 is almost twice as bad as LRU. The performance of the randomized algorithm MARK is better than any deterministic algorithms: it behaves almost 2/3 better than LRU. Observe that LFD, an optimal offline algorithm, is only

a factor of 3 better than LRU under this measure. Contrast this with the competitive ratio for which LFD is k times better than LRU.

The adaptive and bijective models generalize to other online problems. In the interests of brevity we refer the reader to Appendix where we give to other examples in which adaptive analysis leads to more accurate results.

5 Classical Measures

In this section we discuss the basic features of the classical model with the aim of understanding the differences between the various models. Classical analysis of algorithms uses primarily the worst-case measure to quantify the performance of an algorithm. We briefly review the underlying assumptions and highlight certain key aspects of the measure. Recall that the driving motivation for algorithm analysis is to understand, quantify and compare the relative performance of a set of algorithms for a given problem. Traditionally, there has been a preference for measures which return a single numerical value. Such a measure would naturally induce a linear order in the set of algorithms thus allowing for ready comparisons between them. However, as we know, for the case of algorithms in general it is not possible to reduce their performance down to a single number. Hence we must consider measures that are functions of one (or more) parameters, and introduce a method for making pairwise comparisons.

Formally, given an algorithm $\mathcal{A}$ and an input x, $\mathcal{A}(x)$ denotes the execution of $\mathcal{A}$ over x. A performance measure μ associates a real positive number to $\mathcal{A}(x)$. Typical examples of μ are time, space usage, number of random bits used, and number of I/O operations.

When analyzing an algorithm we aim to determine the value of μ for a given input in as accurate a manner as feasible. Observe that the domain of $\mu(\cdot)$, which consists of all binary strings representing well formed inputs lacks regularity. This makes it difficult to evaluate or estimate $\mu(\mathcal{A}(x))$ for a given input x. Hence, we group inputs by size. This draws on the familiar notion developed in elementary school that for a fixed type of problem—say addition or multiplication—its complexity increases with the number of digits in the input. In other words this assumption in the model is at some level naturally supported by experience.

With the grouping by size assumption in place we can now define a new measure—termed a timing function—from the positive integers to the real numbers, i.e. $T : \mathbb{Z}^+ \leftarrow \mathbb{R}^+$. The function $T(n)$ is defined as

$$T(n) = \max_{|x|=n} \{T_A(x)\}$$

where $T_A(x) = \mu(A(x))$ denotes the time taken by algorithm $\mathcal{A}$ to run on input x.

While this measure has come to be seen as synonymous to the performance of an algorithm it is important to remember that it is in fact an imprecise approximation.

Indeed, the amount of information lost in this reduction varies significantly with the algorithm. We illustrate this with three algorithms: Mergesort, Bubblesort with early termination, and Quicksort. Figures 1, 2, and 3, show the performance of these algorithms over a billion random permutations of 100 numbers. Each dot in the tear drop shape represents the number of comparisons required by an actual input among the billion permutations. We can see that the performance for Mergesort over most inputs is strongly clustered around 525 ± 10 comparisons—which is reasonably close the predicted worst-case value of 664 comparisons (see Figure 1). The actual observed worst case for Mergesort over a billion permutations was 554 comparisons. For Bubblesort with early ter-

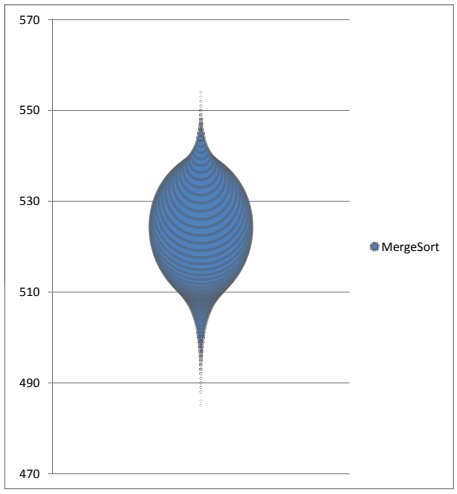

Figure 1: Performance of Mergesort over a billion random permutation of 100 numbers.

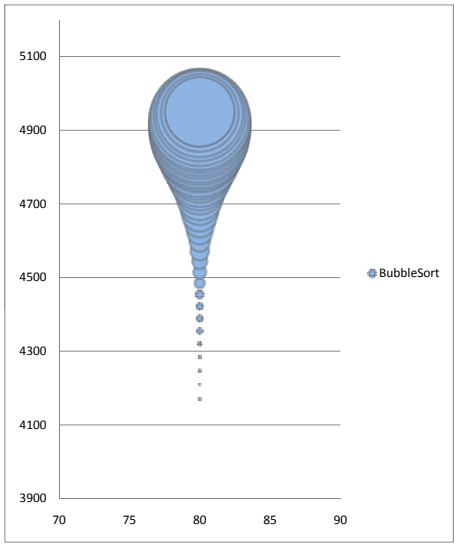

Figure 2: Performance of Bubblesort over a billion random permutation of 100 numbers.

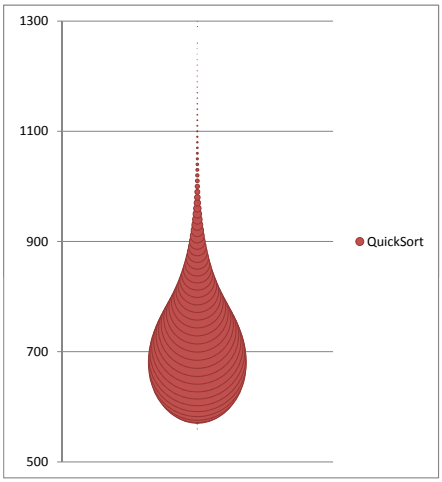

Figure 3: Performance of Quicksort over a billion random permutation of 100 numbers.

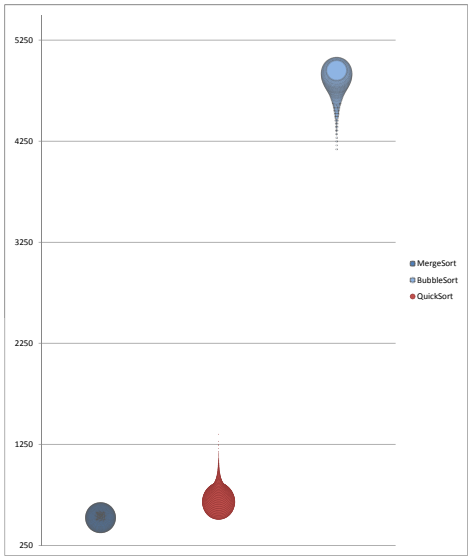

Figure 4: Relative performance of sorting algorithms over a billion random permutation of 100 numbers.

mination we have a similar situation where the performance is strongly clustered around 4920 ± 20 comparisons which is very close to the worst case performance of 4950 comparisons (see Figure 2). Moreover the worst case bound of 4450 was observed in practice for several of the permutations generated at random. Lastly we have Quicksort. For this algorithm we can see that the worst case performance is a terrible representative its actual performance. Quicksort timings are strongly clustered around 700 ± 40 comparisons but the worst case prediction is 4950 comparisons! Even the observed worst case over a billion permutations which was on itself very rare, required 1427 comparisons, falling well short of the predicted worst case.

While this is well known it illustrates that worst case performance is, in all three cases, just an approximation of the actual behaviour we were interested in studying and that the quality of this approximation is variable. Note that the fact that this bound on the performance has been mathematically proven is orthogonal to the quality of this representation. The example given also illustrates the ways in which a model or measure can fail and yet still be the preferred tool. In Section 6 we will discuss this in more detail.

Observe that in the definition of worst case analysis there are several choices to be made. First we have the choice of measure. For example, by replacing the time measure function μ with a function μ' measuring space or I/O operations we obtain the standard worst case space and I/O-model analysis, respectively. Second we have the choice of the worst case max operator. We could use instead the average operator which gives average case analysis for any of the measures listed above. Lastly we have the grouping by input size. There are alternatives to this grouping though none as commonly used as those for the previous two choices. For polynomial algorithms there is adaptive analysis which, as discussed before, uses a measure of difficulty to group inputs by a second dimension. For example, in Table 4 we list the set of input permutations to a comparison based sort. Each column is grouped by the standard measure of input size but further subdivided by the complexity measure of transpositions required to obtaine the sorted sequence. Observe that trivially all sequences in the first few k rows, for k constant, can be sorted in time $O(n)$. For NP-hard problems we have parameterized analysis, which uses a set of measures—such as treewidth—to group the inputs by a parameter other than input size. Another example is given by output-sensitive algorithms in which the input space is partitioned by both the size of the input and the size of the output produced by a particular input string.

Probability Measures. The partition of the input space by size is motivated by the goal of introducing structure to the input space as well as obtaining a mapping from the reals to the reals, since real valued functions are more amenable to analysis. Note that this situation parallels the use of random variables in probability. In this case the probability space often is similarly not amenable to study due to its lack of structure and hence the notion of random variable is introduced. A random variable transforms the space of events (which could as varied as a set of coin tosses or a sample subset of people) into classes which are represented by the pre-image of their numerical value, i.e. the set of events $X^{-1}(r)$ for $r \in R$.

In practice, in nearly all instances the random variable X is some sort of natural counting function, yet the formal definition chooses to ignore this fact and allows for any arbitrarily defined random variable. Contrast this with the analysis of algorithms in which we selected the most natural grouping function (input size). This grouping has only relatively recently been expanded to include a second parameter such as difficulty of the input, treewidth or output size. At the same time it has become increasingly clear that there is a large number of such grouping functions which

input size	1	2	3	4
0 transpositions	1	1 2	1 2 3	1 2 3 4
1 transposition		2 1	1 3 2 2 1 3	1 2 4 3 1 3 2 4 2 1 3 4
2 transpositions			2 3 1 3 1 2	1 3 4 2 1 4 2 3 2 1 4 3 2 3 1 4 3 1 2 4
3 transpositions			3 2 1	1 4 3 2 2 3 4 1 2 4 1 3 3 1 4 2 3 2 1 4 4 1 2 3
4 transpositions			3 1 2	2 4 3 1 3 2 4 1 3 4 1 2 4 1 3 2 4 2 1 3
5 transpositions				3 4 2 1 4 2 3 1 4 2 1 3
6 transpositions				4 3 2 1

Table 4: Number of transpositions needed for sorting an input of size n for $n = 1, 2, 3$, and 4.

are natural and worth being the subject of study[3].

Similarly the measurement function Pr which also has a very natural interpretation in mathematics is left unspecified by the axiomatization of Kolmogorov and is only required to be a measurement function assigning the value of one to the entire sample space.

In turn in computer science we selected a specific random-variable-of-sorts namely the input size, and then a specific measure or density function, namely max. Thus far the field has only hesitantly considered variations of those, as discussed before. This is in stark contrast to the probabilistic definition in which no judgement is made as to the specific choice of parameters.

In practice, the most commonly used measures aside from time are space and number of I/O operations. In this it can be argued that while this set might not yet be complete it is unlikely to contain many other functions and that by keeping the set of possible measures small leads to

[3]Indeed, at a recent Dagstuhl meeting on fixed parameter tractability over a dozen different parameters were identified by the attendants as being in common use in the field.

stronger theorems, just as in probability theory often theorems assume a specific type of distribution or at least independent identically distributed variables. So in all, the fact that a narrower choice was made in algorithms analysis is not necessarily bad, but it is important to keep in mind that this choice was made and expand this selection if need be.

Observe that the competitive ratio can also be understood in this light. In its original form it simply computes the maximum of a normalized measure over the entire input space. That is, let r denote the competitive ratio of an online algorithm A then

$$r = \max_{x} \left\{ \frac{cost_A(x)}{cost_{OPT}(x)} \right\}$$

which in particular leads to the initial terming of algorithms for which the competitive ratio was a function of n *non-competitive* or of unbounded competitive ratio. Eventually this definition was enlarged to include a partition of the input space by input size as follows:

$$r(n) = \max_{|x|=n} \left\{ \frac{cost_A(x)}{cost_{OPT}(x)} \right\}$$

which allows us to describe an algorithm as, say $\log n$-competitive or $\sqrt{n}$-competitive.

Following the same outline, we define the adaptive performance of an algorithm A as

$$T(n, \lambda) = \max_{\substack{|x|=n \\ D(x)=\lambda}} \{T_A(x)\}$$

which gives a bivariate timing function. Fixed parameter algorithms, approximation algorithms and output-sensitive algorithms also use a second parameter to refine the partition of the input space. For example, consider the performance of Quicksort as shown in Figure 3. A well chosen difficulty parameter would partition the performance teardrop into narrower ranges each of which would then presumably be estimated much more accurately by its corresponding maximum value.

In sum, classical measures have made certain parameter choices, which while natural and useful are not necessarily unique. In cases where these choices led to undesirable results one can either perfect the classical model, or introduce a novel measure to handle the anomalous cases. Past experience shows that properly chosen variants have led to interesting and novel algorithms and results.

6 Evaluating new models

As observed before, proposing a new model or measure is relatively rare, and widespread adoption of one is even rarer. As such there is a paucity of established techniques on how to gauge the usefulness and validity of a proposed new model. In this section we discuss key aspects of the introduction and evaluation of new models to illustrate the process by which new models become adopted.

Often there is a tendency to think in terms of "competing models" as if there were a unique best model out there waiting to be found. In practice, different models are used in different settings depending on the information being sought and the specific application in mind. For example if we are sorting a given number of keys, we might choose to study the problem under the RAM model, the I/O model or the MapReduce model if the number of keys is modest, large or massive,

respectively. That is to say, to adopt the MapReduce model we do not need to disprove the validity of the RAM model throughout. All that is required is for the MapReduce model to produce more accurate estimates of the algorithm performance for the particular set of instances that we are interested in, and then we use this model for exactly those instances.

Furthermore, newly introduced models often have to be evaluated on their promise as it might take the combined work of several research efforts before their full potential is grasped[4]. Indeed a well-honed and understood inferior model will initially outperform an ultimately preferable new model that is yet to be fully developed and perfected.

Another often overlooked factor is that models are approximations of reality, not accurate descriptions—hence the use of the word model. This was most notably stated by the mathematician George Box in his maxim "essentially, all models are wrong, but some are useful"[5] [BD86, page 424]. In fact models can be at times highly inaccurate provided that they fail in well understood and predictable scenarios, in which case we learn not to use them in the first place. For example, we know that the worst case measure is not adequate for the evaluation of randomized algorithms. Naturally this is not considered an indictement of the worst case measure, and rightly so.

The economist Paul Krugman observed this process of internalization of the limits of well established models and overlooking of their flaws, in the context of commenting on the various models he introduced:

> The models I proposed were incomplete and yet they told meaningful stories. To achieve this aggregate level of description required accepting certain basically silly assumptions of symmetry, yet these silly assumptions tell stories that were persuasive, and that could not be told using the standard model. What I began to realize was that *in economics we are always making silly assumptions; it's just that some of them have been made so often that they come to seem natural.* And so one should not reject a model as silly until one sees where its assumptions lead. Initially your assumptions will surely look peculiar. Consider for example the Arrow-Debreu model of perfect competition with utility maximization and complete markets. This is indeed a wonderful model—not because its assumptions are remotely plausible but because it helps us think more clearly about both the nature of economic efficiency and the prospects for achieving efficiency under a market system. [*emphasis added and excerpted for brevity, see [Kru] for the complete quote*]

This is not to say that newly proposed models cannot be tested or evaluated. A good model shows promise even if initially flawed and should eventually lead to new insights and predictions that were not explicitly built into the model.

A good model first unifies, then explains and ultimately predicts. Ideally it begins by reducing the facts to basic, self-evident principles which are so obviously true that they verge between the trivial, the overly simplified and the circular.[6] Yet they readily and accurately explain the facts in ways that were not otherwise possible.

[4]Consider for example the introduction of the notion of NP-completeness by S. Cook and the followup paper by Richard Karp the year after, discussing this idea further.

[5]Another variant, also by G. Box, is "remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."

[6]For example, consider Newton's 1st Law: "Objects remain in motion until they are affected by an external force". What is an external force? something that affects the motion of an object.

Observe that the effectiveness of a model is another important consideration. Indeed, we readily trade accuracy for effectiveness in our choice of models. We illustrated this fact with the error observed during the worst case analysis of the sorting algorithms in the previous section. In a more recent example we see the same tradeoff being made in the development of non-uniform access-cost memory models:

> A number of models have been proposed [...] The two most widely adopted ones are the input/output model (or I/O model) and the cache-oblivious model. Their success is due to the balance they provide between simplicity, in order to allow the design and analysis of sophisticated algorithms and accuracy in predicting the performance of algorithms on real memory hierarchies. (*excerpted from [PAZ09]*)

It is easy to underestimate the amount of precision that we routinely trade in exchange for effectiveness. In this sense almost all probabilistic models are a vast tradeoff between accuracy and effectiveness. For example, given an unlimited budget a doctor might be able to determine if a patient will or will not develop lung cancer. Instead we settle for a probabilistic statement such as "30% of former smokers do". In all three cases discussed above, even though we might be able to produce much better estimates it would be at a disproportionately higher cost and hence not done.

7 Conclusions

In this paper, we highlighted the gap between theoretical and experimental results for some online problems and possible ways to close this gap. We observed that standard measure for analysis of online algorithms, i.e., competitive analysis, leads to results that are not consistent with practice for paging. Then we described reasons for the shortcomings of competitive analysis and described several new models for analysis of online algorithms that do not have these drawbacks. Bijective analysis and Average analysis directly compare two online algorithms on all sequences of the same length and lead to satisfactory results when applied to paging. They avoid two shortcomings of competitive analysis: they do not compare online algorithms to an optimal offline algorithm and they do not focus on a single worst case sequence. We then proposed a more effective measure, namely adaptive analysis. We discussed various choices made by established models as well as possible alternatives and contrast it with those made in probability theory. Lastly we discussed how to go about evaluating a newly introduced model or measure.

References

[ADLO07] S. Angelopoulos, R. Dorrigiv, and A. López-Ortiz. On the separation and equivalence of paging strategies. In *Proc. SODA*, pages 229–237, 2007.

[ADLO08] S. Angelopoulos, R. Dorrigiv, and A. López-Ortiz. List update with locality of reference. In *Proc. LATIN*, pages 399–410, 2008.

[AFG05] S. Albers, L. M. Favrholdt, and O. Giel. On paging with locality of reference. *JCSS*, 70(2):145–175, 2005.

[AL08] S. Albers and S. Lauer. On list update with locality of reference. In *Proc. ICALP*, pages 96–107, 2008.

[ALOH08] Spyros Angelopoulos, Alejandro López-Ortiz, and Angèle M. Hamel. Optimal scheduling of contract algorithms with soft deadlines. In *AAAI*, 2008.

[AS09] S. Angelopoulos and P. Schweitzer. Paging and list update under bijective analysis. In *Proc. SODA*, pages 1136–1145, 2009.

[AV88] A. Aggarwal and J. S. Vitter. The Input/Output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.

[BD86] G. E. P. Box and N. R Draper. *Empirical model-building and response surface*. John Wiley & Sons, Inc., 1986.

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

[Bec04] L. Becchetti. Modeling locality: A probabilistic analysis of LRU and FWF. In *Proc. ESA*, pages 98–109, 2004.

[BEY97] R. Bachrach and R. El-Yaniv. Online list accessing algorithms and their applications: Recent empirical evidence. In *Proc. SODA*, pages 53–62, 1997.

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

[BF03] J. Boyar and L. M. Favrholdt. The relative worst order ratio for on-line algorithms. In *Proc. Italian Conf. on Algorithms and Complexity*, 2003.

[BFL05] J. Boyar, L. M. Favrholdt, and K. S. Larsen. The relative worst order ratio applied to paging. In *Proc. SODA*, pages 718–727, 2005.

[BHH+02] Therese C. Biedl, Masud Hasan, Joseph Douglas Horton, Alejandro López-Ortiz, and Tomás Vinar. Searching for the center of a circle. In *CCCG*, pages 137–141, 2002.

[BIRS95] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *JCSS*, 50:244–258, 1995.

[BL99] J. Boyar and K. S. Larsen. The Seat Reservation Problem. *Algorithmica*, 25(4):403–417, 1999.

[BLN01] J. Boyar, K. S. Larsen, and M. N. Nielsen. The Accommodating Function: A generalization of the competitive ratio. *SIAM Journal on Computing*, 31(1):233–258, 2001.

[BM04] J. Boyar and P. Medvedev. The relative worst order ratio applied to seat reservation. In *Proc. SWAT*, pages 90–101, 2004.

[CN99] M. Chrobak and J. Noga. LRU is better than FIFO. *Algorithmica*, 23(2):180–185, 1999.

[DELO09] R. Dorrigiv, M. R. Ehmsen, and A. López-Ortiz. Parameterized analysis of paging and list update algorithms. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA '09)*, 2009. to appear.

[Den68] P. J. Denning. The working set model for program behaviour. *CACM*, 11(5):323–333, 1968.

[Den80] P. J. Denning. Working sets past and present. *IEEE Transactions on Software Engineering*, SE-6(1):64–84, 1980.

[Den05] P. J. Denning. The locality principle. *CACM*, 48(7):19–24, 2005.

[DHS95] A. Datta, Ch. Hipke, and S. Schuierer. Competitive searching in polygons—beyond generalized streets. In *Proc. 6th ISAAC*, pages 32–41. LNCS 1004, 1995.

[DLO05] R. Dorrigiv and A. López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, September 2005.

[DLO07] R. Dorrigiv and A. López-Ortiz. The cooperative ratio of on-line algorithms. Technical Report CS-2007-39, University of Waterloo, Cheriton School of Computer science, October 2007.

[DLO08] R. Dorrigiv and A. López-Ortiz. Adaptive searching in one and two dimensions. In *Proc. CCCG*, pages 215–218, 2008.

[FW98] A. Fiat and G. J. Woeginger. Competitive odds and ends. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms — The State of the Art*, volume 1442 of *LNCS*, pages 385–394. Springer-Verlag, 1998.

[HH85] J. H. Hester and D. S. Hirschberg. Self-organizing linear search. *ACM Computing Surveys*, 17(3):295, September 1985.

[HIKK01] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM J. Comput*, 31(2):577–600, 2001.

[Ick94] Ch. Icking. *Motion and Visibility in Simple Polygons.* Dissertation, Fernuniversität Hagen, 1994.

[IKM93] Christian Icking, Rolf Klein, and Lihong Ma. How to look around a corner. In *CCCG*, pages 443–448, 1993.

[IKP96] S. Irani, A. R. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25:477–497, 1996.

[Ira98] S. Irani. Competitive analysis of paging. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms — The State of the Art*, volume 1442 of *LNCS*, pages 52–73. 1998.

[Ken96] C. Kenyon. Best-fit bin-packing with random order. In *Proc. SODA*, pages 359–364, 1996.

[KMSY94] M.-Y. Kao, Y. Ma, M. Sipser, and Y. Yin. Optimal constructions of hybrid algorithms. In *Proc. 5th SODA*, pages 372–381, 1994.

[KP00] E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30:300–317, 2000.

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

[KRT93] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proc. 4th SODA*, pages 441–447, 1993.

[Kru] P. Krugman. How i work. Available at: http://web.mit.edu/krugman/www/howiwork.html.

[LO96] A. López-Ortiz. *On-line Target Searching in Bounded and Unbounded Domains*. PhD thesis, Dept. of Comp. Sci., University of Waterloo, 1996.

[LOAH06] Alejandro López-Ortiz, Spyros Angelopoulos, and Angèle M. Hamel. Optimal scheduling of contract algorithms for anytime problems. In *AAAI*, 2006.

[LOS96] A. López-Ortiz and S. Schuierer. Walking streets faster. In *Proc. 5th SWAT*, pages 345–356. LNCS 1097, 1996.

[LOS01a] A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays? *Theoretical Computer Science*, 261(2):267–295, 2001.

[LOS01b] Alejandro López-Ortiz and Graeme Sweet. Parallel searching on a lattice. In *CCCG*, pages 125–128, 2001.

[LOS02] Alejandro López-Ortiz and Sven Schuierer. Online parallel heuristics and robot searching under the competitive framework. In *SWAT*, pages 260–269, 2002.

[LOS04] A. López-Ortiz and S. Schuierer. On-line parallel heuristics, processor scheduling and robot searching under the competitive framework. *Theoretical Computer Science*, 310(1-3):527–537, 2004.

[PAZ09] Chris Hamilton Peyman Afshani and Norbert Zeh. Cache-oblivious range reporting with optimal queries requires superlinear space. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 277–286. ACM, 2009.

[Pro99] H. Prokop. Cache-oblivious algorithms. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1999.

[PS06] K. Panagiotou and A. Souza. On adequate performance measures for paging. In *Proc. STOC*, pages 487–496, 2006.

[RS71] Henry R. Richardson and Lawrence D. Stone. Operations analysis during the underwater search for scorpions. *Naval Research Logistics Quarterly*, 18(2):141–157, 1971.

[RWS94] N. Reingold, J. Westbrook, and D. D. Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11:15–32, 1994.

[Sch98] F. Schulz. Two new families of list update algorithms. In *Proc. ISAAC*, pages 99–108, 1998.

[ST85a] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *CACM*, 28:202–208, 1985.

[ST85b] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *JACM*, 32(3):652–686, 1985.

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

[Uni] New Mexico State University. Homepage of new mexico state university tracebase (online). Available at: http://tracebase.nmsu.edu/tracebase.html.

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

[You98] N. E. Young. Bounding the diffuse adversary. In *Proc. SODA*, pages 420–425, 1998.

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

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

A Applications of New Models

We observed that applying the adaptive and bijective models to the paging problem leads to promising results. In this appendix we show that they can be applied to other online problems.

A.1 List Update

List update is a fundamental problem in the context of online computation. Consider an unsorted list of m items. The input to the algorithm is a sequence of n requests that should be served in an online manner. Let $\mathcal{A}$ be an arbitrary online 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. Three well-known deterministic online 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 [ST85a].

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 [BEY97].

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

Algorithm	Lower Bound	Upper Bound
General	$\widehat{\lambda}$	$m \cdot \widehat{\lambda}$
MTF	$\widehat{\lambda}$	$\widehat{\lambda}$
Transpose	$\frac{m \cdot \widehat{\lambda}}{2}$	$m \cdot \widehat{\lambda}$
FC	$\approx \frac{m \cdot \widehat{\lambda}}{2}$	$m \cdot \widehat{\lambda}$
TS	$\approx 2\widehat{\lambda}$	$m \cdot \widehat{\lambda}$
Bit	$\approx \frac{3}{2}\widehat{\lambda}$	$m \cdot \widehat{\lambda}$

Table 5: The results for list update.

have been proposed [Tor98, AFG05, Bec04]. Likewise, many researchers have pointed out that input sequences of list update algorithms in practice show locality of reference [HH85, Sch98, BEY98] and actually online list update algorithms try to take advantage of this property [HH85, RWS94]. Hester and Hirschberg [HH85] 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 [HH85] 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 [ADLO08, AL08, DLO07].

In [ADLO08], Angelopoulos, Dorrigiv and López-Ortiz extended the concave analysis model [AFG05] 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. [BEY97]). They combined Average analysis with concave analysis and proved that under this model MTF is never outperformed, while it always outperforms *any other online list update algorithm*. Thus, [ADLO08] resolved the open problem posed by Hester and Hirschberg [HH85].

Dorrigiv et al. [DELO09] studied the parameterized complexity of list update algorithms in terms of locality of reference. Here we briefly present their results. We define the non-locality of sequences for list update in an analogous way to the corresponding definition for paging (Definition 1) . The only differences are:

1. We do not normalize the non-locality by the length of the sequence, i.e., $\widehat{\lambda}(\sigma) = \sum_{1 \leq i \leq |\sigma|} d_\sigma[i]$.
2. If $\sigma[i]$ is the first access to an item we assign the value m to $d_\sigma[i]$.[7]

The results are summarized in Table 5. According to these results, MTF has the best performance among well known list update algorithms. TS has performance at least twice as bad as MTF. The performance of Transpose and FC is at least $m/2$ times worse than MTF. The performance of Bit is worse than MTF, while its competitive ratio is better. Experimental results of [BEY97] show that MTF has better performance than Bit in practice, which favors our result over competitive analysis.

[7] As for paging, asymptotically, and assuming the number of requests is much larger than m, any constant can replace m for the $d_\sigma[i]$ of the first accesses.

A.2 Online Motion Planning

Online motion planning has been an active area of research within theoretical computer science for well over a decade. The applications are varied ranging from robotics, to search-and-rescue operations in the high seas [RS71, LOS01b] as well as in land, such as in an avalanche [BHH+02] or an office space [HIKK01, DHS95, DHS95, Ick94], to scheduling of heuristic algorithms for solvers searching an abstract solution space for a specific solution [KMSY94, KRT93, LOS02, ALOH08, LOAH06]. A large number of different search and motion scenarios have been considered and theoretically optimal solutions have been derived for many of them. Yet, few if any of those solutions have led to changes in practical algorithms. As in the case of paging, we first contrast the assumptions of the theoretical model with those in practice.

The theoretical model has traditionally used the competitive ratio framework to quantify the performance on online search algorithms. In contrast, robotics researchers consider simply distance traversed to destination, perhaps augmented with the number of scan operations along the way. The competitive ratio naturally implies then a worst case metric, while systems people care about the everyday metric. For example, a vacuuming robot does not need to be able to efficiently vacuum mazes. It needs, on the other hand, the ability to efficiently vacuum a living room. The model of motion assumed in theory is often continuous curved motion paths, with perfect motion, while in practice piecewise linear motions are often preferable and there is a consistent, large-degree of forward and rotational error present in most cases. Theoretical models often assume perfect scans and other such detection mechanisms, even though in practice most scans are low resolution (although rapidly increasing in quality). In other settings, such as robotic cars driving across an unpaved road, the degree of inaccuracy is high, and is likely to remain so for the foreseeable future given the angles of resolution and distances of travel. Lastly, the competitive ratio assumes scenes built with the purpose of foiling the robot in its task. For office and warehouse applications this is once again an unwarranted and overly pessimistic assumption leading to unreasonable motion planning algorithms in practice. While certain applications such as guarding and securing a polygon against intruders should be studied under adversarial models, others such as search and rescue and warehouse applications should assume friendly users which have an interest in being found and/or making the environment as amenable as possible, within the circumstances, for the robot to succeed. So in the best case we have an actively cooperating target, such as in search and rescue, in which case the proper framework for analysis is the cooperative ratio, or as in the case of a robot navigating a warehouse or office environment, moving obstacles should not be assumed to behave adversarially, and an adaptive analysis is more appropriate.

In this case, the challenge is to define a reasonable adaptive measure that reflects realistic scenes. While in the case of paging locality of reference is a natural adaptive measure, it is not quite so obvious what characteristics describe a realistic scene as opposed to a contrived one. In general "natural" scenes have predominantly orthogonal angles along the walls, which have bounded aspect ratio features. In turn obstacles are smaller, with more general angles and often of unbounded aspect ratio due to the discretization effects of a non rectilinear object. Lastly there is an interesting effect in that there is a certain expectation that the path followed by the robot be predictable and hence wandering is considered an undesirable effect. Practitioners seem to prefer paths with a few sharp moves.

We have initiated the study of certain primitives, which are widely used within the theoretical motion planning community, including looking around the corner and exploring a line (i.e. two semi-infinite rays) in search for a target identifiable upon contact. In the latter an adaptive measure leads

to a more "natural" search path even though surprisingly the actual distance traversed is larger both in the worst and expected case [DLO08]. However it remains an open problem to determine a reasonable and effective adaptive measure for the exploration of all polygonal regions.