A Survey of Performance Measures for On-line Algorithms

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

Abstract

The competitive ratio is the most common metric in on-line algorithm analysis. The growth and strength of the field is due in no small part to the effectiveness of this measure in the course of practical analysis. On the other hand, there are known applications in which the competitive ratio produces somewhat 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. In addition, there are situations in which we might simply desire a different measure than that provided by the competitive ratio. Because of this various alternatives to the competitive ratio have been proposed in the literature. In this paper we survey several of those alternatives, highlight their distinctive properties and discuss their benefits and drawbacks.

1 Introduction

The competitive ratio is the most common metric in on-line algorithm analysis. Formally introduced by Sleator and Tarjan, it has served as a practical framework for the study of algorithms that must make irrevocable decisions in the presence of only partial information [ST85]. The competitive ratio has been a great success, enabling the measurement of the performance of various well known heuristics and fostering the development of the field. On-line algorithms are more often than not amenable to analysis under this framework; that is, computing the competitive ratio has proven to be effective—even in cases where the exact shape of the OPT solution is unknown. As well, it has been successfully applied outside the original on-line paging setting to other applications such as on-line geometric searching and on-line approximation to NP-complete problems.

The competitive ratio metric can be derived from the observation that an on-line algorithm, in essence, computes a partial solution to a problem using incomplete information.

Then, it is only natural to quantify the performance drop due to this absence of information. That is, we compare the quality of the solution obtained by the on-line algorithm with the one computed in the presence of full information, namely that of the off-line optimal OPT.

On the other hand, there are known applications in which the competitive ratio produces somewhat 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. The pessimistic nature of the competitive ratio derives from two aspects, one is the use of the worst case `max` operator, the second is the use of a highly idealized off-line OPT algorithm in the denominator. In addition, there are situations in which we might simply desire a different measure than that provided by the competitive ratio.

Because of this various alternatives to the competitive ratio have been proposed in the literature. In this survey we list eleven well known alternatives to the competitive ratio, highlight their distinctive properties and discuss their benefits and drawbacks.

2 Competitive Ratio

Paging is a classic example of a problem studied in the context of on-line algorithm analysis. A paging algorithm must decide which k memory pages to keep in the cache without the benefit of knowing the *sequence* of upcoming page requests. The goal of a paging algorithm is to minimize the number of cache misses over the entire input sequence. The paging algorithm must produce a partial solution after receiving the ith page request and determine which page to evict, shall the page request be a cache miss. The performance of the paging algorithm is quantified by the number of cache misses, which is the *cost* of a particular solution. For a given paging algorithm, we consider the sequences with the worst (highest) possible cost and compare them to the off-line optimal OPT which knows the entire sequence in advance. Alternatively, since we are maximizing the cost over all input sequences, we can think of an *adversary* which selects the next element in the input sequence in a way that is least desirable for the paging algorithm.

We can generalize the key concepts of this example to other problems as follows. Let $\sigma = (s_1, s_2, \ldots)$ be an input sequence. We denote by $\sigma_{1:j} = (s_1, s_2, \ldots, s_j)$ the prefix subsequence of the first j requests in σ. An on-line algorithm $\mathcal{A}$ for an minimization 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 s_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 if*

$$\mathcal{A}(\sigma) \leq c \cdot \text{OPT}(\sigma)$$

for all input sequences σ.

Some of the early literature considers only algorithms with constant competitive ratio, and all others are termed as algorithms with *unbounded* competitive ratio. Alternatively, we can define a $C(n)$-competitive algorithm as follows

Definition 2 *An on-line algorithm $\mathcal{A}$ is said to have competitive ratio $C(n)$ if, for all input sequences σ we have:*

$$\mathcal{A}(\sigma) \leq C(|\sigma|) \cdot \mathrm{OPT}(\sigma).$$

This definition can be relaxed to describe the asymptotic competitive ratio

Definition 3 *An on-line algorithm $\mathcal{A}$ is said to have asymptotic competitive ratio $C(n)$ if*

$$\mathcal{A}(\sigma) \leq C(n) \cdot \mathrm{OPT}(\sigma) + b \qquad \textit{for all } \sigma.$$

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

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

We note that this definition has three key components: the max operator, the on-line cost in the numerator and the optimal off-line cost in the denominator. To better understand their role, in the next subsection we briefly review three well known performance measures used in classical algorithm analysis. Highlighting the differences between these well known measures will help us understand the motivations behind some of the alternative measures we will describe in Section 3.

2.1 A note on classical performance measures

Let the timing function $T_{\mathcal{A}}(I)$ denote the time taken by an off-line algorithm $\mathcal{A}$ over an input I.

Definition 4 *The* worst-case *time of algorithm $\mathcal{A}$ is defined as*

$$A(n) = \max_{|I|=n} \{T_{\mathcal{A}}(I)\}.$$

Definition 5 *The* average-case *time of $\mathcal{A}$ under an uniform distribution is defined as*

$$\overline{A}(n) = \mathrm{avg}_{|I|=n}\{T_{\mathcal{A}}(I)\} = \frac{\sum_{|I|=n} T_{\mathcal{A}}(I)}{\#\{|I| = n\}},$$

where # denotes the cardinality function.

In the case of a general distribution, the average-case time of $\mathcal{A}$ is

$$\overline{A}(n) = E(T_{\mathcal{A}}(I) \mid |I| = n) = \sum_{|I|=n} T_{\mathcal{A}}(I) Pr(I),$$

where E denotes the expectation function of classical probability. Observe that max, avg, and $E(\cdot)$ are aggregate functions over all inputs of size n.

A third measure, albeit somewhat less common, comes from adaptive analysis. This measure is used on problems in which much simpler instances appear frequently. The idea is to require good performance on all inputs, as compared to only on the worst case or the average case.

Definition 6 *An algorithm $\mathcal{A}$ is said to be* adaptive *with respect to a measure $V(I)$ if $T_{\mathcal{A}}(I) = O(V(I))$ for all input I or equivalently*

$$\max_{\forall I} \left\{ \frac{T_{\mathcal{A}}(I)}{V(I)} \right\} \leq c$$

for some constant c.

This can be generalized to the notion of $f(n, V(I))$-adaptive algorithms which are those such that $T_{\mathcal{A}}(I) = f(n, V(I))$, for all inputs I of length n. Again this can be expressed as a ratio

$$\max_{|I|=n} \left\{ \frac{T_{\mathcal{A}}(I)}{V(I)} \right\} = f(n, V(I)).$$

This expression combines both the max aggregate operator of worst case analysis and a comparison ratio which is reminiscent of competitive analysis.

Observe that the these performance measures have, as in the case of the on-line competitive ratio, three key choices: (i) an aggregate function over inputs, (ii) a numerator with possibly its own aggregation function and in the case of the average and adaptive measures (iii) a denominator with its own aggregation function and involving an external measure.

2.2 On-line performance measures

Consider again the competitive ratio measure $C(n)$:

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

This measure suffers in practice from two aspects. One the denominator is an off-line algorithm that has full knowledge of the future request sequence and unbounded computational power. In certain instances the comparison with such a powerful adversary leads to algorithms of varying degrees of sophistication having the same equally bad competitive ratio. For example an algorithm that has good performance in all but one rather complicated instance has the same competitive ratio as an algorithm that always makes a bad decision (even on "trivial" instances) so long as the bad decisions are never worse than that of the rare worst case of the preferred algorithm.

3 Alternatives to the Competitive Analysis

The competitive ratio is the standard measure for analyzing and studying on-line algorithms. Although it is useful in many cases, it has some shortcomings. Most of these are due to the fact that the competitive ratio is a pessimistic measure. For some on-line algorithms their competitive ratio is much worse than their performance ratios in practice. As well, in some cases, the competitive ratio cannot distinguish between on-line algorithms that behave very differently in practice. For example Least-Recently-Used (LRU) and First-In-First-Out (FIFO) are two on-line paging algorithms that have the same competitive ratios, while in practice LRU is much better than FIFO [Ira98]. Another problem with competitive analysis is that it cannot be used to directly compare two on-line algorithms; we must compare each algorithm to some optimal off-line algorithm. The competitive ratio also has some counter-intuitive features. For example, making the on-line algorithm more powerful by finite look-ahead does not change the competitive ratio or adding resources might worsen the competitive performance of an on-line algorithm [BNS69] (the so called Belady anomaly).

Therefore several alternative measures for the quality of on-line algorithms have been proposed [BDB94, BLN01, BF03, KP00, Ken96, FW98, Ira98, BIRS95, IKP96, CN99, You94]. We study some of these alternatives in this section. The basic problem in competitive analysis is that the adversary is too powerful compared to the on-line algorithm. Many papers try to overcome this problem by restricting the power of the adversary. For example this is usually done by placing some constraints on the set of legal input sequences. Another problem is that a single unusual sequence can result in a very bad competitive ratio that is far from the quality of typical sequences.

3.1 Accommodation Ratio and Accommodating Function

These two measures are the same as the competitive ratio except that they restrict the legal input sequences. They apply to on-line problems with limited resources and only consider those input sequences in which the optimal solution does not benefit from having more than a certain amount of resources. We use an example to explain this. The *fair bin packing* problem consists of n bins of size k and an input sequence I of items where the size of each item is an integer between 1 and k. This sequence is given to the algorithm in an on-line manner and we want to maximize the total number of items in the bins. Also the packing should be fair; we can reject an item only if it cannot fit in any bins when it is given. Although the optimal off-line algorithm knows the whole sequence I in advance, it should *fairly* process the requests in the same order as I.

Now for the accommodating ratio [BL99] we consider those input sequences that can be packed in n bins by a fair optimal off-line algorithm. In general we only consider those input sequences in which the optimal off-line algorithm does not benefit from having more resources than those already available. For the accommodating function [BLN01, BFLN03] we only consider those input sequences that can be packed in αn bins by a fair optimal off-line algorithm, where $\alpha > 0$ is the function's variable.

Next we define these measures more precisely. Consider an on-line problem Π with limited resources. Let $\mathcal{A}(I)$ be the cost of an on-line algorithm $\mathcal{A}$ on an input sequence I and let

$\text{OPT}(I)$ be the cost of an optimal off-line algorithm on I. Let OPT_m be the cost of OPT when an amount m of the limited resource is available.

Definition 7 *[BFLN03] Let P be an on-line problem with a fixed amount n of resources. For any $\alpha > 0$, an input sequence I is said to be an α-sequence, if $\text{OPT}_{\alpha n}(I) = \text{OPT}_{n'}(I)$, for all $n' \geq \alpha n$ (1-sequences are also called accommodating sequences).*

Definition 8 *Let $\mathcal{A}$ be an on-line algorithm for a minimization problem. Then $\mathcal{A}$ is c-competitive on α-sequences, if there exists a constant b, such that $\mathcal{A}(I) \leq c \cdot \text{OPT}(I) + b$, for any α-sequence I. The* accommodating function *is defined as*

$$\mathsf{A}_{\mathcal{A}}(\alpha) = \inf\{c \mid \mathcal{A} \text{ is } c\text{-competitive on } \alpha\text{-sequences}\}$$

Note that accommodation ratio is $\mathsf{A}(1)$. In other words, the accommodation ratio is the same as the competitive ratio when we restrict the input to accommodating sequences (1-sequences). Therefore the accommodation ratio is usually called the competitive ratio on accommodating sequences. Also the competitive ratio is $\lim_{\alpha\to\infty} \mathsf{A}(\alpha)$. Thus the accommodating function is an extension of both the competitive ratio and the accommodation ratio.

Several papers [BL99, BLN99, BBJ+00, BBE+03, EF03, BN99, ABE+02] use the accommodating ratio as the measure of quality for on-line algorithms. Boyar et al. [BL99] give lower bounds and upper bounds for the competitive ratio and accommodating ratio of two versions (unit price and proportional price) of the *seat reservation* problem. They proved these results for BEST-FIT, FIRST-FIT, and general deterministic fair algorithms. The two measures (the competitive ratio and the accommodating ratio), agree on the proportional price problem but differ on the unit price problem. The bounds are tight for the proportional price problem, but not in the unit price problem. Bach et al. [BBJ+00, BBE+03] give better and tight bounds for the deterministic fair algorithms for the unit price problem. They also consider randomized algorithms and prove some bounds for them.

Several algorithms for the on-line fair bin packing problem are analyzed in [BLN98]. Upper bounds and lower bounds are given for the accommodating ratio for FIRST-FIT, WORST-FIT, and general algorithms. The lower bound for FIRST-FIT is improved in [BN99]. According to these bounds, FIRST-FIT has strictly better accommodating ratio than WORST-FIT. However if we consider the standard competitive ratio, it can be shown [BLN99] that WORST-FIT behaves strictly better than FIRST-FIT. Therefore the competitive ratio and the accommodating ratio can give contradictory results. Epstein and Favrholdt [EF03] consider a variation of on-line fair bin packing in which bins can have different sizes and give upper bounds and lower bounds for the accommodating ratio of several on-line algorithms.

The unrestricted bin-packing problem is the same as the fair bin packing problem except that we do not require the algorithms to be fair. Azar et al. [ABE+02] study this variation of the problem and compare it with fair bin packing using the accommodating ratio. They prove an asymptotically tight bound for the accommodating ratio of FIRST-FIT for the fair bin packing problem. They design an on-line algorithm called UNFAIR-FIRST-FIT which has asymptotically better accommodating ratio than FIRST-FIT in the unrestricted bin packing problem. Finally upper bounds on the accommodating ratio of deterministic and randomized algorithms are proven for the unrestricted bin packing problem.

The accommodation function was studied by Boyer et al. for the fair bin packing problem [BLN01]. They prove lower bounds and upper bounds on the accommodation functions of First-Fit, Worst-Fit, and all deterministic fair algorithms for all $\alpha \geq 1$. A variant of the seat reservation problem in which seat changes are allowed is studied in [BKN04]. Lower bounds and upper bounds for the competitive ratio, accommodating ratio, and accommodation function are proven for several algorithms. Finally, Boyer et al. [BFLN03] extend the accommodation function to values of $\alpha < 1$. They study the accommodation function of several algorithms for the seat reservation and unrestricted bin packing problems. For the seat reservation problem, they show that we can separate the performance of three algorithms using the accommodation function at $\alpha = 1/3$, while we cannot do the same using the competitive ratio or the accommodating ratio. They also studied the connection between the accommodation function and the resource augmentation technique [KP95].

3.2 Max/Max Ratio

The Max/Max ratio [BDB94] tries to be more optimistic by comparing the amortized worst case behaviour of the on-line algorithm with the amortized worst case behaviour of the optimal off-line algorithm. Recall that in competitive analysis we compare the two algorithms on the same sequence. However, this approach is sometimes problematic because the existence of only one bad sequence can drastically change the result. Thus this measure tries to avoid the situation in which a single unusual sequence gives a very bad result. This becomes more clear with the following example used in [BDB94] as a motivation for defining the Max/Max ratio. Consider the problem of buying an insurance policy in an on-line manner. It is reasonable to pay \$5 a month to insure your car against theft. However, this is not a competitive strategy because the off-line adversary can select the scenario in which one will never present a claim to the insurance agent. In the Max/Max ratio, we compare the two algorithms on their respective worst case sequences of the same length.

We define this measure more precisely for an on-line minimization problem Π. The definition for maximization problems is similar. Let $\mathcal{A}$ be an algorithm for Π and let $\mathcal{A}(I)$ be the cost of A on an input sequence I.

Definition 9 *The amortized cost of $\mathcal{A}$ is defined as $M(A) = \limsup_{l\to\infty} M_l(\mathcal{A})$ where $M_l(\mathcal{A}) = \max_{|I|=l} \mathcal{A}(I)/l$. The* Max/Max ratio *of $\mathcal{A}$ denoted $w_M(\mathcal{A})$ is*

$$\limsup_{l\to\infty} \frac{M_l(\mathcal{A})}{M_l(\text{OPT})} = \frac{M(\mathcal{A})}{M(\text{OPT})}$$

where OPT *is an optimal off-line algorithm.*

Note that we can directly compare two on-line algorithms $\mathcal{A}$ and $\mathcal{B}$ using this measure because we have $\frac{M(\mathcal{A})}{M(\mathcal{B})} = \frac{w_M(\mathcal{A})}{w_M(\mathcal{B})}$. Also it is shown [BDB94] that, when considering the Max/Max ratio, look-ahead can improve the on-line performance even in cases where the competitive ratio does not improve.

3.3 Random Order Ratio

The random order ratio [Ken96] is another measure that tries to decrease the dependence on some unusual bad sequences. Let $\mathcal{A}$ be an on-line algorithm for an on-line minimization problem and let $\mathcal{A}(I)$ be the cost of $\mathcal{A}$ on an input sequence $I = (i_1, i_2, \ldots, i_n)$.

Definition 10 *The* random order ratio *of $\mathcal{A}$ is defined as*

$$RC(\mathcal{A}) = \limsup_{\mathrm{OPT}(I)\to\infty} \frac{E_\sigma \mathcal{A}(I_\sigma)}{\mathrm{OPT}(I)}$$

where σ is a permutation of $\{1, 2, \ldots, n\}$, I_σ is the permuted sequence $(i_{\sigma_1}, \ldots, i_{\sigma_n})$, and the expectation is taken over all permutations of $\{1, 2, \ldots, n\}$.

Therefore this measure assumes that all orderings of an input sequence are equally likely. This is a reasonable assumption for some on-line problems. [Ken96] contains a lower bound and an upper bound on the random order ratio of BEST-FIT algorithm for on-line bin-packing problem which are better bounds than their corresponding bounds for the competitive ratio. However, it seems that this measure is difficult generalize to other on-line problems.

3.4 Relative Worst Order Ratio

The relative worst order ratio [BF03, BM04, BFL05] combines some desirable properties of the Max/Max ratio and the random order ratio. Using this measure we can directly compare two on-line algorithms. Informally, for a given sequence it considers the worst case ordering of that sequence for each algorithm and compares their behaviour on these orderings. Then it finds among all sequences (not just reorderings) the one that maximizes the worst case performance. Thus this measure can be considered as a modification of the Max/Max ratio in that we consider the worst sequence among those which are permutations of each other instead of considering the worst sequence among all those having the same length as the Max/Max ratio does. It is also related to random order ratio as it considers permutations of a sequence. However instead of taking the expectation of the algorithm's behaviour on all permutations, it considers the permutation with the worst behaviour.

Let $\mathcal{A}$ and $\mathcal{B}$ be on-line algorithms for an on-line minimization problem and let $\mathcal{A}(I)$ be the cost of $\mathcal{A}$ on an input sequence $I = (i_1, i_2, \ldots, i_n)$. Denote by I_σ the sequence obtained by applying a permutation σ to I, i.e. $I_\sigma = (i_{\sigma_1}, \ldots, i_{\sigma_n})$. Define $\mathcal{A}_W(I) = \min_\sigma \mathcal{A}(I_\sigma)$.

Definition 11 *[BFL05] Let $S_1(c)$ and $S_2(c)$ be the statements about algorithms $\mathcal{A}$ and $\mathcal{B}$ defined in the following way.*

$$S_1(c): \textit{ There exists a constant } b \textit{ such that } \mathcal{A}_W(I) \leq c \cdot \mathcal{B}_W(I) + b \textit{ for all } I.$$

$$S_2(c): \textit{ There exists a constant } b \textit{ such that } \mathcal{A}_W(I) \geq c \cdot \mathcal{B}_W(I) - b \textit{ for all } I.$$

The relative worst order ratio $WR_{\mathcal{A},\mathcal{B}}$ of an on-line algorithm $\mathcal{A}$ to algorithm $\mathcal{B}$ is defined if $S_1(1)$ or $S_2(1)$ holds. In this case $\mathcal{A}$ and $\mathcal{B}$ are said to be comparable. If $S_1(1)$ holds, then $WR_{\mathcal{A},\mathcal{B}} = \sup\{r | S_2(r)\}$, and if $S_2(r)$ holds then $WR_{\mathcal{A},\mathcal{B}} = \inf\{r | S_1(r)\}$.

$WR_{\mathcal{A},\mathcal{B}}$ can be used to compare the qualities of $\mathcal{A}$ and $\mathcal{B}$. If $WR_{\mathcal{A},\mathcal{B}} = 1$ then these two algorithms have the same quality with respect to this measure. The magnitude of difference between $WR_{\mathcal{A},\mathcal{B}}$ and 1 reflects the difference between the behaviour of the two algorithms. For a minimization problem, $\mathcal{A}$ is better than $\mathcal{B}$ with respect to this measure if $WR_{\mathcal{A},\mathcal{B}} < 1$ and vice versa.

The idea behind this measure is that some on-line algorithms perform well on some types of sequence orderings and other algorithms perform well on some other types of orderings. Therefore certain algorithms that cannot be compared using competitive analysis may be comparable in this measure. Boyar and Favrholdt showed that the relative worst order ratio is transitive [BF03].

Note that we can also compare the on-line algorithm $\mathcal{A}$ to an optimal off-line algorithm OPT. The *worst order ratio* of $\mathcal{A}$ is defined as $WR_{\mathcal{A}} = WR_{\mathcal{A},\mathrm{OPT}}$. For some problems, OPT is the same for all order of requests on a given input sequence and hence the worst order ratio is the same as the competitive ratio. However for other problems such as fair bin packing the order does matter for OPT.

In [BM04], three on-line algorithms (First-Fit, Best-Fit, and Worst-Fit) for two variants of the seat reservation problem [BL99] are compared using the relative worst order ratio. All of these three algorithms can be compared in this framework while this is not possible within the classical competitive analysis framework. The relative worst order ratio is applied to paging algorithms in [BFL05]. It is shown that LRU is strictly better than FWF with respect to the worst order ratio, while these two algorithms have the same competitive ratio. Also a new paging algorithm, Retrospective-LRU (RLRU), is proposed and it is shown that RLRU is better than LRU with respect to the relative worst order ratio. This contrasts with results on the competitive ratio of these algorithms. It is also shown that look-ahead is helpful when we consider the relative worst order ratio.

3.5 Loose Competitiveness

Loose competitiveness was first proposed in [You94] and later modified in [You02]. We describe it for an on-line minimization problem. It attempts to obtain a more realistic measure by considering the following two aspects in the analysis of the on-line algorithms. First, in many real on-line problems, we can ignore those input sequences on which the on-line algorithm incurs a cost less than a certain threshold. Second, many on-line problems, have a second resource parameter (e.g. size of cache, number of servers) and the input sequences are independent of these parameters. In contrast, in competitive analysis the adversary can select sequences tailored against those parameters. We clarify this situation by considering the paging problem. In this case the problem parameter is the size in pages k of the cache. Consider the following lower bound on the competitive ratio of any deterministic paging algorithm.

Theorem 1 *[ST85] The competitive ratio of any deterministic on-line paging algorithm is at least k.*

This result can be easily proven by considering an adversary that selects only $k + 1$ pages and at each time requests a page that is not in the cache. For this to work the adversary

needs to know the problem parameter k, . However, in practice the competitive ratios of many on-line paging algorithms have been observed to be constant [You02], i.e. independent of k. This can be obtained by applying loose competitiveness.

In loose competitiveness we consider an adversary that is oblivious to the parameter by requiring it to give a sequence that is bad for most values of the parameter rather than just a specific bad value of the parameter. Let $\mathcal{A}_k(I)$ denote the cost of an algorithm $\mathcal{A}$ on an input sequence I, when the parameter of the problem is k.

Definition 12 *[You02] An algorithm $\mathcal{A}$ is (ϵ, δ)-loosely c-competitive if, for any input sequence I and for any n, at least $(1-\delta)n$ of the values $k \in \{1, 2, \ldots, n\}$ satisfy*

$$\mathcal{A}_k(I) \leq \max\{c \cdot \text{OPT}_k(I), \epsilon\,|I|\}.$$

Therefore we ignore the input sequences I which costs less than $\epsilon\,|I|$. Also we require the algorithm to be good for at least $(1-\delta)$ fraction of the possible parameters. For each on-line problem, we can select the appropriate constants ϵ and δ. The following result shows that by this modification of the competitive analysis, we can obtain paging algorithms with constant performance ratios.

Theorem 2 *[You02] Every k-competitive paging algorithm is (ϵ, δ)-loosely c-competitive for any $0 < \epsilon, \delta < 1$, and $c = (e/\delta)\ln(e/\epsilon)$, where e is the base of the natural logarithm.*

3.6 Access Graph Model

The access graph model was introduced by Borodin et al. to solve two main problems in the competitive analysis of on-line paging algorithms [BIRS95]. One of these problems is that the practical performance ratio of LRU is much better than its competitive ratio. We have mentioned the second problem before: Although LRU and FIFO have the same competitive ratio, LRU behaves much better than FIFO in practice. One reason that LRU has good experimental behaviour is that in practice page requests show *locality of reference*. Temporal locality means that when a page is requested it is more likely to be requested in the near future. Spatial locality means that when a page is requested it is more likely it that a nearby page will be requested in the near future.

In the access graph model we weaken the adversary by restricting the set of legal input sequences. This is done by restricting the set of pages that can be requested after each page. More specifically, we have an access graph $G = (V, E)$ so that each vertex v represents a page p_v and there is an edge from a vertex u to a vertex v if and only if p_v can be requested after p_u. This graph can be directed on undirected depending on the actual practical problem. Locality of reference can be imposed in this model because when we request a page p we should request p or one of its neighbours in the next step. The competitive ratio is the same as in standard competitive analysis except that we restrict ourselves to the input sequences that conform to the given access graph.

Using this model, several interesting results can be obtained [BIRS95, IKP96, CN99, FR97]. For every graph G and every number k of pages in the fast memory, let $c_k(G)$ denote the best competitive ratio that can be achieved by an on-line paging algorithm. Borodin

et al. proved that computing $c_k(G)$ is computable for every finite access graph G [BIRS95]. They also show how to compute the competitive ratio of LRU for every access graph and every k within a factor of two (plus additive constant), and . propose a simple algorithm that nearly achieves the best competitive ratio for every access graph. This algorithm, called FAR, evicts, on each fault, the unmarked page in cache whose distance from a marked page is maximum in the access graph. They proved that the competitive ratio of FAR for every undirected access graph and every k is within $O(\log k)$ of the best possible competitive ratio. This was later improved by Irani et al. showed that the competitive ratio of FAR is $O(c_k(G))$ for any undirected graph G [IKP96]. Experimental results show that some variations of FAR algorithm behave better than LRU in practice [BIRS95]. It is also known that the competitive ratio of LRU is at least as good as FIFO on every access graph [CN99].

3.7 Diffuse Adversary Model

The diffuse adversary model [KP00] tries to refine the competitive ratio by decreasing the power of the adversary. It does this by restricting the set of legal input sequences. Recall that in standard competitive analysis we do not put any restriction on the input sequences and so they can have any distribution. In other words, the on-line algorithm knows nothing about the distribution of the input sequences. At the other end of the extreme, in classic probabilistic analysis of on-line algorithms, the exact distribution of input sequences is known to the on-line algorithm. In the diffuse adversary model, the on-line algorithm does not know the exact distribution, but it knows that it is a member of a class Δ of distributions.

Definition 13 *Let $\mathcal{A}$ be an on-line algorithm for a minimization problem and let Δ be a class of distributions for the input sequences. Then $\mathcal{A}$ is c-competitive against Δ, if there exists a constant b, such that*

$$\mathcal{E}_{I \in D}\mathcal{A}(I) \leq c \cdot \mathcal{E}_{I \in D}\mathrm{OPT}(I) + b,$$

for every distribution $D \in \Delta$, where $\mathcal{A}(I)$ denotes the cost of $\mathcal{A}$ on the input sequence I and the expectations are taken over sequences that are picked according to D.

In other words the adversary selects the distribution D in Δ that is the worst distribution for A. If Δ is more restricted then A knows more about the distribution of input sequences and the power of adversary is more constrained. When Δ contains all possible distributions then the competitive analysis against Δ is the same as the standard competitive ratio. Therefore the diffuse adversary model is an extension of standard competitive analysis. Note that we can also model locality of reference using the diffuse adversary model by considering only those distributions that are consistent with the given access graph. This means that if there is no edge between the vertices corresponding to two pages, the probability that one page is accessed after the other should be zero in our distributions.

This model is applied to the paging algorithms [KP00] by considering a class Δ_ϵ of distributions and proving that LRU has the best competitive ratio against Δ_ϵ among all deterministic on-line algorithms. For any sequence ρ of pages and any page p, let $\mathcal{P}(p|\rho)$ denote the probability that p is the next page requested provided that the request sequence seen so

far is ρ. For any $0 \leq \epsilon \leq 1$, Δ_ϵ contains distributions in which $\mathcal{P}(p|\rho) \leq \epsilon$ for every page p and every page sequence ρ. Computing the actual competitive ratio of both deterministic and randomized algorithms against Δ_ϵ is studied in [You98, You00]. An estimation of the optimal competitive ratio for several algorithms (such as LRU and FIFO) within a factor of 2 is given. Also it is observed that around the threshold $\epsilon \approx 1/k$, the best competitive ratios against Δ_ϵ are $\theta(\ln k)$. The competitive ratios rapidly become constant for values of ϵ less than the threshold. For $\epsilon = \omega(1/k)$, i.e. values greater than the threshold, the competitive ratio rapidly tends to $\theta(k)$ for deterministic algorithms while it remains unchanged for randomized algorithms.

3.8 Smoothed Competitiveness

Some algorithms that have very bad *worst case* performance behave very well in practice. One of the most famous examples is the simplex method. This algorithms has a very good performance in practice but it has exponential worst case running time. *Average case* analysis of algorithms can somehow explain this behaviour but sometimes there is no basis to the assumption that the inputs to an algorithm are random.

Smoothed analysis of algorithms [ST04] tries to explain this intriguing behavior without assuming anything about the distribution of the input instances. In this model, we randomly perturb (smoothen) the input instances according to a probability distribution f and then analyze the behavior of the algorithm on these perturbed (smoothed) instances. For each input instance $\check{I}$ we compute the neighborhood $N(\check{I})$ of $\check{I}$ which contains the set of all perturbed instances that can be obtained from $\check{I}$. Then we compute the expected running time of the algorithm over all perturbed instances in this neighborhood. The smoothed complexity of the algorithm is the maximum of this expected running time over all the input instances. Intuitively, an algorithm with a bad worst case performance can have a good smoothed performance if its worst case instances are isolated. Spielman and Teng show [ST04] that the simplex algorithm has polynomial smoothed complexity. Several other results are known about the smoothed complexity of the algorithms [BMB03, MR05, BD02, ST03].

As we said before, the competitive analysis is a very pessimistic measure and an algorithm can have a very bad competitive ratio only because of a few bad input sequences. Therefore the competitive ratio is a reasonable choice for applying smoothed analysis. This was first done by Becchetti et al. [BLMS+03] who introduced *smoothed competitive analysis*. Informally, smoothed competitive analysis is the same as the competitive analysis except that we consider the cost of the algorithm on randomly perturbed adversarial sequences. As in the analysis of the randomized on-line algorithms, we can have either an *oblivious adversary* or an *adaptive adversary*. The smoothed competitive ratio of an on-line algorithm $\mathcal{A}$ for a minimization problem can be formally defined as follows.

Definition 14 *[BLMS+03] The* smoothed competitive ratio *of an algorithm $\mathcal{A}$ is defined as*

$$c = \sup_{\check{I}} \mathcal{E}_{I \leftarrow N(\check{I})}[\frac{\mathcal{A}(I)}{OPT(I)}],$$

where the supremum is taken over all input instances $\check{I}$, and the expectation is taken over

all instances I that are obtainable by smoothening the input instance $\check{I}$ according to f in the neighborhood $N(\check{I})$.

Note that it is also possible to define the smoothed competitive ratio as

$$c = \sup_{\check{I}} \frac{\mathcal{E}_{I \leftarrow N(\check{I})}[\mathcal{A}(I)]}{\mathcal{E}_{I \leftarrow N(\check{I})}[OPT(I)]}.$$

In [BLMS+03], the first definition is used but it is remarked that a similar result can be obtained using the second definition. They used the smoothed competitive ratio to analyze the MULTI-LEVEL FEEDBACK(MLF) algorithm for processor scheduling in a time sharing multitasking operating system. This algorithm has very good practical performance but its competitive ratio is very bad. They obtain strictly better ratios using the smooth competitive analysis than with the competitive ratio.

3.9 Search Ratio

The search ratio belongs to the family of measures in which the off-line OPT is weakened. It is defined only for the specific case of geometric searches in an unknown terrain for a target of unknown position. Recall that the competitive ratio compares against an all knowing OPT, Indeed, for geometric searches in the competitive ratio framework, the OPT is simply a shortest path algorithm, while the on-line search algorithm has intricate methods for searching. The search ratio instead considers the case where OPT knows the terrain but not the position of the target. That is, the search ratio compares two search algorithms, albeit one more powerful than the other. By comparing two instances of like objects the search ratio can be argued to be a more meaningful measure of the quality of an on-line search algorithm. Koutsopias et al. show that searching in trees results the same large competitive ratio regardless of the algorithm, yet under the search ratio framework certain algorithms are far superior to others [KPY96].

3.10 Travel Cost

As we observed in Section 2, classical complexity time analysis generally uses an unnormalized time measure even though a normalized measure has been defined and proven fruitful in certain settings. This raises up the possibility of using an unnormalized cost measure for on-line algorithms as well. This measure has been used in on-line geometric searches, in which the main objective is to minimize the length of the longest search sequence, known as the travel cost of the solution. Formally

Definition 15 *The* travel cost *of an on-line algorithm $\mathcal{A}$ on input I is given by*

$$C(n) = \max_{|I| \leq n} \{A(I)\}$$

For example in the case of an actual search and rescue operation in the high seas minimizing the maximum search time is more relevant than the competitive ratio on any particular point in the search path [FSBY+04].

3.11 Average Ratio

The average competitive ratio was introduced in the context of competitive searches on the real line for a target of unknown location. The classic problem in this field is the cow path problem. As traditionally described, a cow reaches a fork on the road and recalls that in one and only one of the two paths there is a pasture field. This problem was first analyzed under the competitive ratio and its solution predates the introduction of the competitive ratio in on-line algorithms. The optimal solution under the standard competitive ratio metric is 9-competitive. However, on the average, the pasture is discovered at an average competitive ratio of approximately 4.59, assuming a uniform distribution of the position of the field. Interestingly enough, the strategy resulting in the optimal average cost is different from the optimal one under the competitive ratio framework [Gal79, LO96]. Formally we have,

Definition 16 *The average competitive ratio, or* average ratio *for short is defined as*

$$E_{\forall\,|I|}\left(\frac{A(I)}{\mathrm{OPT}(I)}\right).$$

4 Conclusions

In this survey we presented eleven alternative measures to the competitive ratio for the analysis of on-line algorithms. This list, while not exhaustive, is illustrative of the various different approaches to improve on the competitive ratio.

5 Acknowledgments

We thank Spyros Angelopoulos for many helpful discussions on alternative performance measures for on-line algorithms.

References

[ABE+02] Yossi Azar, Joan Boyar, Leah Epstein, Lene M. Favrholdt, Kim S. Larsen, and Morten N. Nielsen. Fair versus Unrestricted Bin Packing. *Algorithmica*, 34(2):181–196, 2002.

[BBE+03] Eric Bach, Joan Boyar, Leah Epstein, Lene M. Favrholdt, Tao Jiang, Kim S. Larsen, Guo hui Lin, and Rob Van Stee. Tight bounds on the competitive ratio on accommodating sequenced for the seat reservation problem. *Journal of Scheduling*, 6(2):131–147, 2003.

[BBJ+00] Eric Bach, Joan Boyar, Tao Jiang, Kim S. Larsen, and Guo-Hui Lin. Better Bounds on the Accommodating Ratio for the Seat Reservation Problem. In *Sixth Annual International Computing and Combinatorics Conference*, volume 1858 of *Lecture Notes in Computer Science*, pages 221–231. Springer-Verlag, 2000.

[BD02] Avrim Blum and John Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *Proceedings of the 13th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA-02)*, pages 905–914, 2002.

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

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

[BFL05] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. The Relative Worst Order Ratio Applied to Paging. In *Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 718–727. ACM Press, 2005.

[BFLN03] Joan Boyar, Lene M. Favrholdt, Kim S. Larsen, and Morten N. Nielsen. Extending the Accommodating Function. *Acta Informatica*, 40(1):3–35, 2003.

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

[BKN04] Joan Boyar, Susan Krarup, and Morten N. Nielsen. Seat reservation allowing seat changes. *Journal of Algorithms*, 52(2):169–192, 2004.

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

[BLMS+03] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schafer, and T. Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In IEEE, editor, *Proceedings: 44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003, 11–14 October 2003, Cambridge, Massachusetts*, pages 462–471, 2003.

[BLN98] Joan Boyar, Kim S. Larsen, and Morten N. Nielsen. The accommodating function – a generalization of the competitive ratio. Technical Report PP-1998-24, Department of Mathematics and Computer Science, Odense University, December 22 1998. Mon, 6 Nov 2000 09:44:21 GMT.

[BLN99] Joan Boyar, Kim S. Larsen, and Morten N. Nielsen. Separating the accommodating ratio from the competitive ratio. Technical Report PP-1999-03, Department of Mathematics and Computer Science, Odense University, January 25 1999.

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

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

[BMB03] Cyril Banderier, Kurt Mehlhorn, and Rene Beier. Smoothed analysis of three combinatorial problems. In *MFCS: Symposium on Mathematical Foundations of Computer Science*, volume 2747, pages 198–207, 2003.

[BN99] Joan Boyar and Morten N. Nielsen. An improved lower bound on first-fit's accommodating ratio for the unit price bin packing problem. Technical Report PP-1999-11, Department of Mathematics and Computer Science, Odense University, June 10 1999.

[BNS69] L. A. Belady, R. A. Nelson, and G. S. Shedler. An anomaly in space-time characteristics of certain programs running in a paging machine. *Communications of the ACM*, 12(6):349–353, June 1969.

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

[EF03] Leah Epstein and Lene M. Favrholdt. On-line maximizing the number of items packed in variable-sized bins. *Acta Cybernetica*, 16(1):57–66, 2003.

[FR97] Amos Fiat and Ziv Rosen. Experimental studies of access graph based heuristics: Beating the LRU standard? In *Proc. 8th Symp. on Discrete Algorithms (SODA)*, pages 63–72. ACM/SIAM, 1997.

[FSBY+04] Ariel Felner, Roni Stern, Asaph Ben-Yair, Sarit Kraus, and Nathan Netanyahu. PHA*: Finding the shortest path with A* in an unknown physical environment. *Journal of Artificial Intelligence Research*, 21:631–670, 2004.

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

[Gal79] Shmuel Gal. Search games with mobile and immobile hider. *SIAM Journal of Control and Optimization*, 17:99–122, 1979.

[IKP96] Sandy Irani, Anna R. Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM J. Comput.*, 25:477–497, 1996.

[Ira98] Sandy 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. Springer-Verlag, 1998.

[Ken96] Claire Kenyon. Best-fit bin-packing with random order. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.

[KP95] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science*, pages 214–221, 23–25 October 1995.

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

[KPY96] Elias Koutsoupias, Christos Papadimitriou, and Mihalis Yannakakis. Searching a fixed graph. In *International Colloquium on Automata, Languages, and Programming*, volume 1099, pages 280–289, 1996.

[LO96] Alejandro López-Ortiz. *On-line target searching in bounded and unbounded domains*. PhD thesis, University of Waterloo, 1996.

[MR05] Bodo Manthey and Rüdiger Reischuk. Smoothed analysis of the height of binary search trees. Schriftenreihe der Institute für Informatik und Mathematik A-05-17, Universität zu Lübeck, Lübeck, June 2005.

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

[ST03] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of termination of linear programming algorithms. *Mathematical Programming*, 97(1–2):375–404, 2003.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

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

[You98] Neal E. Young. Bounding the diffuse adversary. In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 420–425, 1998.

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

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