Alternatives to competitive analysis
Georgios D Amanatidis

1   Introduction

Competitive analysis allows us to make strong theoretical statements about the performance of an algorithm without making probabilistic assumptions on the inputs. However, it has been often criticized with respect to its realism. A wide range of online algorithms –both good and not so good in practice– seem to have the same competitive ratio. A concrete example of this is the inability of competitive analysis to distinguish between the LRU and the FIFO algorithms for the *paging problem*, whose performances differ markedly in practice. (Recall that, on a fault, LRU evicts the page in memory whose most recent access was earliest, while FIFO evicts the page that has been in memory the longest).

Moreover, competitive analysis frustrates not only interesting algorithms, but also information regimes. The classical example is again from paging. Suppose we allow a paging algorithm to have a lookahead of ℓ steps. Then, although we would expect better performance, the algorithm behaves equally badly as if it had no lookahead, as we'll see later. The search for alternatives to competitive ratio is mainly driven by these weaknesses.

The basic source of this problem is that the best offline algorithm is too powerful compared to any online algorithm. Different alternatives can be exploited to resolve this. The two obvious ways to generalize is to restrict the possible input distributions or/and the class of algorithms we compare the given online algorithm with.

Before moving to such generalizations, recall the definition of competitive ratio of a minimization problem Π and of an algorithm for Π (for maximization problems the definitions are analogous). Algorithm A is c-competitive if there exists a constant d, such that for any input x

$$A(x) \leq c \cdot OPT(x) + d,$$

where OPT denotes the optimum offline algorithm. The competitive ratio of the algorithm A is the infimum of all such c's, and the competitive ratio of Π, R_Π, is the minimum competitive ratio achievable by an online algorithm.

In what follows we present three alternatives to competitive analysis and give some interesting relative results.

2   The Diffuse Adversary Model

It is true in general that the distribution of the inputs is not known. However, competitive analysis assumes we know absolutely nothing about the distribution, which seems a little bit unrealistic in practice. So, the following generalization, *the diffuse adversary model* (defined in [1]), arises quite naturally, by removing the assumption that nothing is known about the input distribution.

Let Π be a minimization problem, and Δ be a known class of possible distributions. We say that a deterministic online algorithm A is c-competitive against the class Δ if there exists a constant d such that for all distributions $D \in \Delta$,

$$\mathbb{E}_D(A(x)) \leq c \cdot \mathbb{E}_D(OPT(x)) + d,$$

where $A(x)$ is the cost of A when presented with input x, and OPT denotes the optimum offline algorithm.

The competitive ratio against Δ of the algorithm A is the infimum of all such c's, and the competitive ratio against Δ of Π, $R_\Pi(\Delta)$, is the minimum competitive ratio against Δ achievable by an online algorithm (we will drop the index Π if there is no chance of confusion).

That is, the adversary picks a distribution D among those in Δ, so that the expected (under D) performance of any online algorithm and the offline optimum algorithm, are as far apart as possible.

Notice that a "worst case distribution" in competitive analysis is just a worst case input with probability one. Therefore, if Δ contains such a distribution we get $R = R(\Delta)$. In particular, if Δ is the class of all possible distributions then $R = R(\Delta)$. Hence, the diffuse adversary model is indeed a refinement of competitive analysis.

Now, let's restrict ourselves to paging. Then the input distribution specifies, for each page a and sequence of page requests σ, the probability $\mathbb{P}(a|\sigma)$ that the next page fault is a given that the sequence so far is σ. It is unrealistic to assume that we know this distribution precisely. However, we can make the simple and natural assumption that the next page request is not predictable with absolute certainty.

So, for any ε between 0 and 1, we define the class Δ_ε to be the class of all distributions obeying the inequality $\mathbb{P}(a|\sigma) \leq \varepsilon$, for all a and σ. A natural question to ask now is, What online algorithm achieves the optimal competitive ratio against Δ_ε, $R(\Delta_\varepsilon)$? The answer, given in [1], is quite interesting. Here we state it without proof:

Theorem 1 *For any ε, LRU has optimal competitive ratio $R(\Delta_\varepsilon)$ for the paging diffuse adversary model.*

It is an open problem to determine the competitive ratio against Δ_ε of known paging algorithms, in particular to estimate the competitive ratio of FIFO. Papadimitriou and Koutsoupias conjecture that FIFO is suboptimal for some values of ε. If this is the case, it will add extra validity to the diffuse adversary model, in the sense that the model can actually distinguish between LRU and FIFO.

3   Comparative Analysis

Comparative analysis was also introduced in [1]. Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of algorithms for the same minimization problem. The *comparative ratio* $R(\mathcal{A}, \mathcal{B})$ is defined as follows:

$$R(\mathcal{A}, \mathcal{B}) = \max_{B \in \mathcal{B}} \min_{A \in \mathcal{A}} \max_x \frac{A(x)}{B(x)}.$$

Notice that the definition does not require $\mathcal{A} \subseteq \mathcal{B}$, although this is usually the case, and this is how we are going to use it here. In particular, notice that if $\mathcal{A}$ is the class of online algorithms and $\mathcal{B}$ is the class of all (online and offline) algorithms for the problem in question, then $R(\mathcal{A}, \mathcal{B}) = R$, that is the comparative ratio becomes the competitive ratio. Hence, comparative analysis is indeed a refinement of competitive analysis.

Before we see an application of comparative analysis, let's take a deeper look in the motivation for the above definition. As discussed in the introduction, we may consider online algorithms for paging with a lookahead of ℓ steps. Obviously, lookahead is a valuable feature of a paging algorithm

in practice. However, it is easy to see that in competitive analysis it doesn't give any power at all. To see this, let A_ℓ be the variation of algorithm A with a lookahead of ℓ steps and $\sigma = \sigma_1\sigma_2\sigma_3\ldots$ be a worst case sequence for A. Consider $\sigma' = \sigma_1^{\ell+1}\sigma_2^{\ell+1}\sigma_3^{\ell+1}\ldots$ and notice that A_ℓ, given σ', faults as often as A, given σ.

The question about the power of lookahead is a question about classes of algorithms and not just a single algorithm. Let's formalize this question. An online algorithm for the paging problem has lookahead ℓ if it can base its decision not only on the past, but also on the next ℓ requests. Let $\mathcal{L}_\ell$ be the class off all the online algorithms for paging with lookahead ℓ. In particular, $\mathcal{L}_0$ is the class of all traditional online algorithms. We want to determine $R(\mathcal{L}_0, \mathcal{L}_\ell)$. The following theorem is due Koutsoupias and Papadimitriou [1].

Theorem 2 *If k is the number of pages in the memory, we have $R(\mathcal{L}_0, \mathcal{L}_\ell) = \min\{\ell + 1, k\}$.*

Proof. Let $n = \min\{\ell, k-1\}$ and let $B \in \mathcal{L}_\ell$. We may assume, without loss of generality, that algorithm B moves pages only to service requests. We are going to prove that algorithm A below has comparative ratio $n + 1$. This proves that $n + 1$ is an upper bound for the comparative ratio $R(\mathcal{L}_0, \mathcal{L}_\ell)$.

So, let A be the following generalization of LRU:

> If a request r is already in the fast memory, do nothing.
> Otherwise, to service r do the following:
>> Ignore the n most recent distinct requests(includind r), and
>> amongst the remaining pages choose to evict one, so that the
>> resulting configuration is as close as possible to the last
>> known configuration of B.

It suffices to show that for every $n + 1$ consecutive page faults of A, there is at least one page fault for B. By using induction on the number of requests we are going to show that after each subsequence of requests that causes c page faults to A but no page fault to B, the configurations of A and B differ by at most $n - c$ pages. In particular, after each subsequence of requests that causes n page faults to A but no page fault to B, both algorithms are in the same configuration. Then it follows that for every $n + 1$ consecutive page faults of A, B suffers at least one page fault, as desired.

Fix a request sequence $\rho = r_1 r_2 \ldots$ and let $A_0, A_1, \ldots$ and $B_0, B_1, \ldots$ be the configurations of A and B that service ρ. Notice that $A_0 = B_0$ and therefore the base of the induction is trivial.

Assume that the induction hypothesis holds for $t - 1$. For the inductive step we consider the following cases.

First consider $r_t \in A_{t-1}$. Then, A suffers no page fault and therefore $|A_t - B_t| \leq |A_{t-1} - B_{t-1}|$. The inductive step follows.

Similarly, when the request $r_t \notin B_{t-1}$, B suffers a page fault and $|A_t - B_t \leq |A_{t-1} - B_{t-1}|$, which is at most n by the induction hypothesis.

Assume now that $r_t \in B_{t-1}$, but $r_t \notin A_{t-1}$. Let x_t be the page evicted by A to service r_t. It is easy to see that if $x_t \notin B_{t-1}$, then $|A_t - B_t| = |A_{t-1} - B_{t-1}| - 1$. The final and more complicated case is when $x_t \in B_{t-1}$. By the definition of algorithm A, x_t is not one of the n most recent requests and consequently x_t is also in B_{t-n}. It follows that $A_{t-1} \subseteq B_{t-n} \cup \{r_{t-n+1}, \ldots, r_{t-1}\}$, otherwise A would not choose $x_t \in B_{t-n}$ to evict. Therefore,

$$A_t \subseteq B_{t-n} \cup \{r_{t-n+1}, \ldots, r_{t-1}, r_t\}.$$

We also have the obvious relation

$$B_t \subseteq B_{t-n} \cup \{r_{t-n+1}, \ldots, r_{t-1}, r_t\},$$

since we assumed that B moves pages only to service requests. If B suffered no page fault during the last c requests, i.e., $B_t = B_{t-1} = \ldots = B_{t-c}$, the set $B_{t-n} \cup \{r_{t-n+1}, \ldots, r_{t-1}, r_t\}$ has cardinality at most $k+n-c$. We conclude that $|A_t \cup B_t| \leq k+n-c$, which implies the desired $|A_t - B_t| \leq n-c$.

To show that $\min\{\ell + 1, k\}$ is also a lower bound of the comparative ratio, let B be a variant of the optimal offline algorithm adapted to lookahead ℓ. More precisely, B never evicts one of the next n requests. Fix a set of $k+1$ pages. Then, for every request sequence ρ, B suffers at most one page fault for every $n+1$ consecutive requests. The lower bound follows because for any algorithm A, there is a request sequence ρ such that A suffers a page fault for every request.

$\square$

4 The Accommodating Function

In this generalization of competitive ratio (defined in [2]) we restrict the set of input sequences that the adversary can provide. We will give the definition of the accommodating function for maximization problems; for minimization problems is analogous. Consider an online problem with a fixed amount of resources n. For an online algorithm A, let $A(I)$ be the value of running A on input I, and $OPT(I)$ be the maximum value that can be achieved on I by an optimal offline algorithm. A and OPT use the same amount of resources n. For a problem with some limited resource, let $OPT_m(I)$ denote the value of an optimal offline algorithm on input I when the amount m of the resource is available. Similarly we may define $A_m(I)$.

Now we give the key definition. Let I be an input sequence. We say that I is an α-sequence if $OPT_{\alpha n}(I) = OPT_m(I)$ for all $m \geq \alpha n$. That is, α-sequences are the input sequences where the optimal offline algorithm will not obtain a better result by having more than the amount αn of resources.

Now, we define the accommodating function $\mathcal{A}(\alpha)$ to be the usual competitive ratio restricted on α-sequences. To be more precise, an algorithm A is c-competitive an α-sequences if there exists a constant d such that for every n and every α-sequence I we have $A_n(I) \geq c \cdot OPT_n(I) - d$. The accommodating function of A is then the supremum of such c's, i.e.

$$\mathcal{A}_A(\alpha) = \sup\{c|\ A \text{ is } c\text{-competitive on } \alpha\text{-sequences}\},$$

while the accommodating function for the problem is

$$\mathcal{A}(\alpha) = \sup_A \mathcal{A}_A(\alpha).$$

Notice that as α goes to infinity, there is no restriction on the input sequences, and therefore the limit $\lim_{\alpha \to \infty} \mathcal{A}(\alpha)$ is exactly the competitive ratio R.

It would be interesting to notice here that the accommodating function is also a generalization of yet another alternative to competitive ratio. In [3] Boyar and Larsen define the accommodating ratio which is the competitive ratio on 1-sequences, that is just $\mathcal{A}(1)$.

In [2], Boyar ,Larsen and Nielsen, showed that the accommodating function provides extra information, by showing that the best choice between First-Fit and Worst-Fit algorithms for Fair

Bin Packing cannot be made based on either the competitive ratio or the accommodating ratio alone. In general, the choice as to which online algorithm to use, depends on which ratio $\mathcal{A}(\alpha)$ is more relevant in a specific situation. This of course depends on the actual distribution of request sequences and on the accommodating function for the algorithms.

Fair Bin Packing. Consider the following online variant of bin packing. Let n be the number of bins, all of size k. Given a sequence of integer sized items of size at most k, we want to maximize the number of items in the bins. Since the problem is online, the requests occur in a definite order. Moreover, we require the packing to be *fair*, that is an item can be rejected only if it cannot fit in any bin at the time it is given.

    Notice that the fairness criterion is part of the problem specification. Therefore, the optimal offline algorithm must process the requests in the same order as the online algorithm and do so fairly, even though it knows the whole sequence in advance.

    Here, the resource is the number of bins; thus, for a given α, we only consider sequences which could be packed in αn bins by an optimal offline algorithm. Next, we prove an upper bound on the accommodating function for fair bin packing:

Theorem 3 *If $k \geq 3$, then for any algorithm for fair bin packing, the accommodating function is at most*

$$
\mathcal{A}(\alpha) = \begin{cases} \frac{6}{7} & \text{if } \alpha = 1 \\[2mm] \frac{2}{2+(\alpha-1)(k-2)} & \text{if } 1 < \alpha \leq \frac{5}{4} \\[2mm] \frac{8}{6+k} & \text{if } \alpha > \frac{5}{4} \end{cases} .
$$

Proof.

$\square$

    Now we'll state without proof the result discussed above. Consider these two algorithms for fair bin packing: *First-Fit*, that places an item in the lowest numbered bin in which it fits, and *Worst-Fit*, that places an item in a bin which is least full. Then we have the theorem below that points out the fact that the choice of an online algorithm depends on which ratio is more relevant in a specific situation; and this in turn depends on the actual distribution of request sequences.

Theorem 4 *If $\mathcal{A}_{FF}$ and $\mathcal{A}_{WF}$ are the accommodation functions for First-Fit and Worst-Fit respectively, then $\mathcal{A}_{FF}(1) > \mathcal{A}_{WF}(1)$, while $\lim_{\alpha \to \infty} \mathcal{A}_{FF}(\alpha) < \lim_{\alpha \to \infty} \mathcal{A}_{WF}(\alpha)$. That is, First-Fit has a better accommodating ratio, while Worst-Fit has a better competitive ratio.*

References

[1] Elias Koutsoupias and Christos H. Papadimitriou, *Beyond Competitive Analysis*, SIAM J. Comput.,30 (2000), pp. 300317.

[2] 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.

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

[4] Allan Borodin, Prabhakar Raghavan, Sandy Irani, Baruch Schieber, *Competitive paging with locality of reference*, STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing :249-259

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