BEYOND COMPETITIVE ANALYSIS

ELIAS KOUTSOUPIAS* AND CHRISTOS PAPADIMITRIOU†

Abstract. The competitive analysis of on-line algorithms has been criticized as being too crude and unrealistic. We propose refinements of competitive analysis in two directions: The first restricts the power of the adversary by allowing only certain input distributions, while the other allows for comparisons between information regimes for on-line decision-making. We illustrate the first with an application to the paging problem; as a byproduct we characterize completely the work functions of this important special case of the k-server problem. We use the second refinement to explore the power of lookahead in server and task systems.

Key words. On-line algorithms, competitive analysis, paging problem, metrical task systems

AMS subject classifications. 68Q05, 68Q25

1. Introduction. The area of *On-Line Algorithms* [14, 10] shares with Complexity Theory the following characteristic: Although its importance cannot be reasonably denied (an algorithmic theory of decision-making under uncertainty is of obvious foundational significance and practical relevance), certain aspects of its basic premises, modeling assumptions, and results have been widely criticized with respect to their realism and relation to computational practice. In this work we revisit some of the most often-voiced criticisms of *competitive analysis* (the basic framework within which on-line algorithms have been heretofore studied and analyzed), and propose and explore some better-motivated alternatives.

In competitive analysis, the performance of an on-line algorithm is compared against an all-powerful adversary on a worst-case input. The *competitive ratio* of a problem—the analog of worst-case asymptotic complexity for this area—is defined as

$$R = \min_A \max_x \frac{A(x)}{\text{opt}(x)} \tag{1.1}$$

Here A ranges over all on-line algorithms, x over all "inputs", opt denotes the optimum off-line algorithm, while $A(x)$ is the cost of algorithm A when presented with input x. This clever definition is both the weakness and strength of competitive analysis. It is a strength because the setting is clear, the problems are crisp and sometimes deep, and the results often elegant and striking. But it is a weakness for several reasons. First, in the face of the devastating comparison against an all-powerful off-line algorithm, a wide range of on-line algorithms (good, bad, and mediocre) fare equally badly; the competitive ratio is thus not very informative, fails to discriminate and to suggest good approaches. Another aspect of the same problem is that, since a worst-case input decides the performance of the algorithm, the optimal algorithms are often unnatural and impractical, and the bounds too pessimistic to be informative in practice. Even enhancing the capabilities of the on-line algorithm in obviously desirable ways (such as a limited lookahead capability) brings no improvement to the ratio (this is discussed more extensively later). The main argument for competitive analysis over the classical approach (minimize the expectation) is that *the distribution*

* Computer Science Department, University of California, Los Angeles, CA 90095 (elias@cs.ucla.edu). Research supported in part by NSF grant CCR-9521606.

† Computer Science Division, University of California, Berkeley, CA 94720 (christos@cs.berkeley.edu). Research supported in part by the National Science Foundation.

is usually not known. However, competitive analysis takes this argument way too far: It assumes that *absolutely nothing* is known about the distribution, that *any distribution* of the inputs is in principle possible; the worst-case "distribution" prevailing in competitive analysis is, of course, a worst-case input with probability one. Such complete powerlessness seems unrealistic to both the practitioner (we always know, or can learn, *something* about the distribution of the inputs) and the theoretician of another persuasion (the absence of a prior distribution, or some information about it, seems very unrealistic to a probabilist or mathematical economist).

The *paging problem,* perhaps the most simple, fundamental, and practically important on-line problem, is a good illustration of all these points. An unreasonably wide range of deterministic algorithms (both the good in practice LRU and the empirically mediocre FIFO) have the same competitive ratio—k, the amount of available memory. Even algorithms within more powerful *information regimes*—for example, any algorithm with lookahead $\ell > 0$ pages—provably can fare no better. Admittedly, there have been several interesting variants of the framework that were at least partially successful in addressing some of these concerns. *Randomized* paging algorithms have more realistic performance [5, 11, 13]. Some alternative approaches to evaluating on-line algorithms were proposed in [1, 12] for the general case and in [2, 6, 7, 15] specifically for the paging problem.

In this paper we propose and study two refinements of competitive analysis which seem to go a long way towards addressing the concerns expressed above. Perhaps more importantly, we show that these ideas give rise to interesting algorithmic and analytical problems (which we have only begun to solve in this paper).

Our first refinement, *the diffuse adversary model*, removes the assumption that we know nothing about the distribution—without resorting to the equally unrealistic classical assumption that we know all about it. *We assume that the actual distribution D of the inputs is a member of a known class Δ of possible distributions.* That is, we seek to determine, for a given class of distributions Δ, the performance ratio

$$R(\Delta) = \min_A \max_{D \in \Delta} \frac{\mathcal{E}_D(A(x))}{\mathcal{E}_D(\mathrm{opt}(x))} \tag{1.2}$$

That is, the adversary picks a distribution D among those in Δ, so that the expected, under D, performance of the algorithm and the off-line optimum algorithm are as far apart as possible. Notice that, if Δ is the class of all possible distributions, (1.1) and (1.2) coincide since the worst possible distribution is the one that assigns probability one to the worst-case input and probability zero everywhere else. Hence the diffuse adversary model is indeed a refinement of competitive analysis.

In the paging problem, for example, the input distribution specifies, for each page a and sequence of page requests ρ, $\mathrm{Prob}(a|\rho)$—the probability that the next page fault is a, given that the sequence so far is ρ. It is unlikely that an operating system knows this distribution precisely. On the other hand, it seems unrealistic to assume that any distribution at all is possible. For example, suppose that the next page request *is not predictable with absolute certainty:* $\mathrm{Prob}(a|\rho) \leq \epsilon$, for all a and ρ, where ϵ is a real number between 0 and 1 capturing the inherent uncertainty of the request sequence. This is a simple, natural, and quite well-motivated assumption; call the class of distributions obeying this inequality Δ_ϵ. An immediate question is, what on-line algorithm achieves the optimal competitive ratio $R(\Delta_\epsilon)$?

As it turns out, the answer is quite interesting. The optimum on-line algorithm is *robust*—that is, the same for all ϵ's—and turns out to be a familiar algorithm that is also very good in practice: LRU. It is noteworthy that LRU emerges from the analysis as the unique "natural" optimal algorithm, although there are other algorithms that may also be optimal. An important byproduct of our analysis is that, extending the work in [9], we completely characterize the work functions of the paging special case of the k-server problem.

In a preliminary version of this work that appeared in [8], we incorrectly stated that the competitive ratio $R(\Delta_\epsilon)$ is given by a simple Markov chain of $(k+\frac{1}{\epsilon})^{k-1}$ states. In fact, the answer is more complicated. The competitive ratio is given by the "optimal" Markov chain from a family of $k^{(k+\frac{1}{\epsilon})^{k-1}}$ Markov chains of $(k+\frac{1}{\epsilon})^{k-1}$ states each.

The second refinement of competitive analysis that we are proposing deals with the following line of criticism: In traditional competitive analysis, the all-powerful adversary frustrates not only interesting algorithms, but also powerful *information regimes.* The classical example is again from paging: In paging the best competitive ratio of any on-line algorithm is k. But what if we have an on-line algorithm *with a lookahead of ℓ steps*, that is, an algorithm that knows the immediate future? It is easy to see that any such algorithm must fare equally badly as algorithms without lookahead. In proof, consider a worst case request sequence $abdc\cdots$ and take its $(\ell+1)$*-stuttered version*, $a^{\ell+1}b^{\ell+1}d^{\ell+1}c^{\ell+1}\cdots$ It is easy to see that an algorithm with lookahead ℓ is as powerless in the face of such a sequence as one without a lookahead. Once more, the absolute power of the adversary blurs practically important distinctions. Still, lookahead is obviously a valuable feature of paging algorithms. How can we use competitive analysis to evaluate its power? Notice that this is not a question about the effectiveness of a single algorithm, but about *classes of algorithms*, about the power of *information regimes*—ultimately, about the value of information.

To formulate and answer this and similar questions we introduce our second refinement of competitive analysis, which we call *comparative analysis.* Suppose that $\mathcal{A}$ and $\mathcal{B}$ are classes of algorithms—typically but not necessarily $\mathcal{A} \subseteq \mathcal{B}$; that is, $\mathcal{B}$ is usually a broader class of algorithms, a more powerful information regime. The *comparative ratio* $R(\mathcal{A},\mathcal{B})$ is defined as follows:

$$R(\mathcal{A},\mathcal{B}) = \max_{B\in\mathcal{B}} \min_{A\in\mathcal{A}} \max_{x} \frac{A(x)}{B(x)} \tag{1.3}$$

This definition is best understood in terms of a game-theoretic interpretation: $\mathcal{B}$ wants to demonstrate to $\mathcal{A}$ that it is a more powerful class of algorithms. To this end, $\mathcal{B}$ proposes an algorithm B among its own. In response, $\mathcal{A}$ comes up with an algorithm A. Then $\mathcal{B}$ chooses an input x. Finally, $\mathcal{A}$ pays $\mathcal{B}$ the ratio $A(x)/B(x)$. The larger this ratio, the more powerful $\mathcal{B}$ is in comparison to $\mathcal{A}$. Notice that, if we let $\mathcal{A}$ be the class of on-line algorithms and $\mathcal{B}$ the class of all algorithms—on-line or off-line—then equations (1.1) and (1.3) coincide, and $R(\mathcal{A},\mathcal{B}) = R$. Hence comparative analysis is indeed a refinement of competitive analysis.

We illustrate the use of comparative analysis by attacking the question of the power of lookahead in on-line problems of the "server" type: If $\mathcal{L}_\ell$ is the class of all algorithms with lookahead ℓ, and $\mathcal{L}_0$ is the class of on-line algorithms, then we show that, in the very general context of *metrical task systems* [3] we have

$$R(\mathcal{L}_0,\mathcal{L}_\ell) = 2\ell+1,$$

(that is, the ratio is at most $2\ell+1$ for all metrical task systems, and it is exactly

$2\ell + 1$ for some), while in the more restricted context of paging

$$R(\mathcal{L}_0, \mathcal{L}_\ell) = \min\{\ell + 1, k\}.$$

2. Diffuse Adversaries. The competitive ratio for a diffuse adversary[1] is given in equation (1.2). In order to make the analysis independent of the initial conditions, we shall allow an additive constant in the numerator. More precisely, a deterministic on-line algorithm A is c-competitive against a class Δ of input distributions if there exists a constant d such that for all distributions $D \in \Delta$:

$$\mathcal{E}_D(A(x)) \leq c \cdot \mathcal{E}_D(\text{opt}(x)) + d \tag{2.1}$$

The competitive ratio of the algorithm A is the infimum of all such c's. Finally, the competitive ratio $R(\Delta)$ of the class of distributions is the minimum competitive ratio achievable by an on-line algorithm. It is important to observe that Δ is a class of acceptable conditional probability distributions; each $D \in \Delta$ is the distribution of the relevant part of the world conditioned on the currently available information. One can easily extend the definition to randomized on-line algorithms. However, in this work, we deal only with deterministic on-line algorithms.

In the case of the paging problem with a set of pages M, Δ is a set of probability distributions on page sequences M^*. An equivalent and perhaps more natural way to describe Δ is by a set of conditional probablity distributions, that is, functions of the form $D : M^* \times M \mapsto [0,1]$, where for all $\rho \in M^*$ $\sum_{a \in M} D(a|\rho) \leq 1$; the sum may be less than 1 because the advesrary may choose to end the sequence. In the game-theoretic interpretation, as the sequence of requests ρ develops, the adversary chooses the values of $D(a|\rho)$ from those available in Δ to maximize the ratio. Since we deal with deterministic algorithms, the adversary knows precisely the past decisions of A, but the adversary's choices may be severely constrained by Δ. It is indicative of the power of the diffuse adversary model that most of the proposals for a more realistic competitive analysis are simply special cases of it. For example, the locality of reference in the paging problem [2, 6] is captured by the diffuse adversary model where Δ consists of the following conditional probability distributions: $D(a|\rho b) = 0$ if there is *no* edge from b to a in the *access graph* and $D(a|\rho b) = 0$ or 1 otherwise. Similarly, the Markov paging model [7] and the statistical adversary model [12] are also special cases of the diffuse adversary model. In the first case, the class Δ of distributions contains only one conditional distribution D with $D(\cdot|\rho b) = D(\cdot|b)$, and in the latter case $D \in \Delta$ assigns probability 1 to some request sequence that satisfies the statistical adversary restriction.

In this section we apply the diffuse adversary model to the paging problem. We shall focus on the class of distributions Δ_ϵ, which contains all functions $D : M^* \times M \mapsto [0, \epsilon]$—that is to say, all conditional distributions with no value exceeding ϵ. But first, we diverge in order to give a useful characterization of work functions for the paging problem.

2.1. The structure of paging work functions. Since the paging problem is the k-server problem on uniform metric spaces, certain key concepts from the k-server theory will be very useful (see [9, 4] for a more detailed exposition).

DEFINITION 2.1. *Let k denote the number of page slots in fast memory, and let M be the set of pages. A* configuration *is a k-subset of M; we denote the set*

[1]Diffuse adversaries are not related to the diffusion processes in probability theory which are continuous path Markov processes.

of all configurations by C. The work function *associated with a sequence $\rho \in M^*$, w_ρ (or simply w when ρ is not important or understood from context) is a function $w_\rho : C \mapsto \Re^+$, defined as follows: $w_\rho(X)$ is the optimum number of page replacements when the sequence of requests is ρ and the final memory configuration is X.*

Notice that in some cases, for example when a configuration X does not contain the last request, some page replacements may occur not because of page faults but because we insist that the final configuration is X.

Henceforth we use the symbols $+$ and $-$ to denote set union and set difference respectively. Also, we represent unary sets with their element, e.g. we write a instead of $\{a\}$. Finally, we denote the cardinality of a set S by $|S|$.

DEFINITION 2.2. *If w is a work function, define the* support *of w to be all configurations $X \in C$ such that there is no $Y \in C$, different from X, with $w(X) = w(Y) + |X - Y|$.*

Intuitively, the values of w on its support completely determine w: if S is the support then $w(X) = \min_{Y \in S}\{w(Y) + |X - Y|\}$. Furthermore, we can safely assume that the optimal off-line algorithm (or adversary) is in a configuration of the support. To see this consider two configurations X and Y with $w_\rho(X) = w_\rho(Y) + |X - Y|$. Then the optimal off-line algorithm has no advantage being in configuration X after servicing ρ instead of configuration Y; it can move from Y to X without exceeding the cost of being at X. In other words, an optimal off-line algorithm which replaces a page only when it is necessary is always in a configuration of the support.

The following lemmata, specific to the paging problem and not true in general for the k-server problem, characterize all possible work functions by determining the structure of their support. A similar, but more complicated, characterization is implicit in the work of [11]. The first lemma states that all configurations in the support have the same value, and hence what matters is the support itself, not the values of w on it.

LEMMA 2.3. *The support of a work function w contains only configurations on which w achieves its minimum value.*

Proof. Towards a contradiction assume that there is a configuration A in the support of w such that $w(A) > \min_X w(X)$. Choose now a configuration B with $w(A) > w(B)$ that minimizes $|B - A|$. By the quasiconvexity condition in [9], there is $a \in A - B$ and $b \in B - A$ such that

$$w(A) + w(B) \geq w(A - a + b) + w(B - b + a).$$

Since $w(A) > w(B)$, this holds only if either $w(A) > w(A - a + b)$ or $w(A) > w(B - b + a)$. In the first case, we get that $w(A) = w(A - a + b) + 1$ and this contradicts the assumption that A is in the support of w. The second case also leads to a contradiction since it violates the choice of B with minimum $|B - A|$, because $|(B - b + a) - A| = |B - A| - 1$. ☐

Since the configuration of the optimal off-line algorithm is always in the support, Lemma 2.3 shows that the off-line cost to service a request is simply the increase (0 or 1) of the minimum value of a work function. As a result, it can be determined on-line when the adversary has a page fault.

The next lemma determines the structure of the support.

LEMMA 2.4. *For each work function w there is an increasing sequence of sets $S_1 \subsetneq S_2 \subsetneq \ldots \subsetneq S_k$, with $S_1 = \{r\}$ the most recent request, such that the support of w is precisely*

$$\{X : |X \cap S_j| \geq j \text{ for all } j\}.$$

We postpone the proof of the lemma to introduce some new definitions. We will call a k-tuple $S = (S_1, S_2, \ldots, S_k)$ of the lemma a *signature* of w. A signature of w completely determines its support. But it doesn't have to be unique. For example, a work function whose support consists of only one configuration $\{a_1, \ldots, a_k\}$ has $k!$ signatures (for each permutation σ let $S_j = \{a_{\sigma_1}, \ldots, a_{\sigma_j}\}$). We define the *type* of w to be the k-tuple $\mathbb{P} = (|S_1| = 1, |S_2|, \ldots, |S_k|)$. Although a work function may have many signatures, its type is unique; it is easy to see that a work function has more than one signatures only when there is a $j > 1$ such that $|S_i| = i$ for $1 \leq i \leq j$.

We now prove Lemma 2.4.

Proof of Lemma 2.4. The proof is by induction on the length of the request sequence. The basis case is obvious: Let $S_j = \{a_1, \ldots, a_j\}$, where $\{a_1, \ldots, a_k\}$ is the initial configuration. For the induction step assume that we have a signature $(S_1, S_2, \ldots, S_k)$ of w, and let w' be the resulting work function after request r.

Consider first the case that r belongs to S_t and not to S_{t-1} for some $t \in \{2, 3, \ldots, k\}$. Since there is at least one configuration in the support of w that contains the request r, the minimum value of w' is the same with the minimum value of w. Therefore, the support of w' is a subset of the support of w. It consists of the configurations that belong to the support of w and contain the most recent request r. It is easy now to verify that w' has the followng signature:

$$\begin{aligned} &S'_1 = \{r\}; \\ &S'_i = S_{i-1} + r, \qquad 2 \leq i \leq t; \\ &S'_i = S_i, \qquad t < i \leq k. \end{aligned}$$

If, on the other hand, the request r does not belong to S_k, the minimum value of w' is one more than the minimum value of w. In this case, the support of w' consists of all configurations in the support of w where one point has been replaced by the request r, i.e. a server has been moved to service r. Consequently, a signature of w' is given by:

$$\begin{aligned} &S'_1 = \{r\}; \\ &S'_i = S_i + r, \qquad 1 < i \leq k. \end{aligned}$$

The induction step now follows. $\square$

Note that the converse of the lemma, not needed in the sequel, also holds: Any such tower of S_j's defines a reachable work function, that is, there exists w_ρ for some ρ with signature S.

We define the *canonical ordering* to be a permutation of all pages such that page a precedes page b when a has been requested more recently than b. Furthermore, pages that have not been requested are ordered according to a fixed ordering (for example, the lexicographic ordering). For example, if the canonical ordering after request sequence ρ is $a_1 \cdots a_m$ then ρ has the form $\cdots a_4(a_3|a_2|a_1)^* a_3(a_2|a_1)^* a_2(a_1)^* a_1$. An obvious property of the canonical ordering is that a new request a_j doesn't change the ordering except from the movement of a_j to the front.

It follows from the proof of Lemma 2.4 that if we know the canonical ordering $a_1 a_2 \cdots$ and the type $(p_1, \ldots, p_k)$ of a work function w, we can determine a signature $(S_1, \ldots, S_k)$ of the work function: S_i consists of the first p_i pages of the canonical ordering, i.e., $S_i = \{a_1, \ldots, a_{p_i}\}$. We will call this signature the *canonical signature* of w.

For a type $\mathbb{P} = (p_1 = 1, p_2, \ldots, p_k)$ that corresponds to a canonical signature $S = (S_1, S_2, \ldots, S_k)$, we will denote by $\mathbb{P}_j$ the type that results from a request in $S_{j+1} - S_j$. From the proof of Lemma 2.4, we get

$$\begin{aligned}
&\mathbb{P}_0 = \mathbb{P},\\
&\mathbb{P}_1 = (1, p_1 + 1, p_3, p_4, \ldots, p_k),\\
&\mathbb{P}_2 = (1, p_1 + 1, p_2 + 1, p_4, \ldots, p_k),\\
&\ \vdots\\
&\mathbb{P}_{k-1} = (1, p_1 + 1, p_2 + 1, p_3 + 1, \ldots, p_{k-1} + 1), \text{ and}\\
&\mathbb{P}_k = (1, p_2 + 1, p_3 + 1, \ldots, p_k + 1).
\end{aligned}$$

The first type $\mathbb{P}_0$ corresponds to a request in S_1 and the last on to a request not in S_k. We will also use the following notation: For types $\mathbb{P} = (p_1, p_2, \ldots, p_k)$ and $\mathbb{Q} = (q_1, q_2, \ldots, q_k)$ we write $\mathbb{P} \le \mathbb{Q}$ (or $\mathbb{Q} \ge \mathbb{P}$) if for all $j = 1, 2, \ldots, k$: $p_j \le q_j$. We also write $\mathbb{P} \trianglelefteq \mathbb{Q}$ (or $\mathbb{Q} \trianglerighteq \mathbb{P}$) if for all $j = 1, 2, \ldots, k-1$: $p_j \le q_{j+1}$. The motivation for this notation is the relation between $\mathbb{P}_{k-1}$ and $\mathbb{P}_k$: $\mathbb{P}_{k-1} \trianglerighteq \mathbb{P}_k$. In addition, $\mathbb{P}_0 \ge \mathbb{P}_1 \ge \mathbb{P}_2 \ge \cdots \ge \mathbb{P}_{k-1} \trianglerighteq \mathbb{P}_k$.

We can now state the following crucial property of work functions.

LEMMA 2.5. *Let ρ and ρ' be two request sequences that result in the same canonical ordering. Let $\mathbb{P}$ and $\mathbb{Q}$ be the types of the two work functions $w = w_\rho$ and $w' = w_{\rho'}$, respectively. If $\mathbb{P} \le \mathbb{Q}$ then any configuration in the support of w is also in the support of w'. Furthermore, if $\mathbb{P} \trianglelefteq \mathbb{Q}$ then for any configuration in the support of w there is configuration in the support of w' that differs in at most one position.*

Proof. Let $S = (S_1, S_2, \ldots, S_k)$ and $S' = (S'_1, S'_2, \ldots, S'_k)$ be the canonical signatures of w and w'. The first case of the lemma, when $\mathbb{P} \le \mathbb{Q}$, is trivial since $S_j \subset S'_j$, $j = 1, 2, \ldots, k$. For the second case, when $\mathbb{P} \trianglelefteq \mathbb{Q}$, it suffices to handle the "largest" possible $\mathbb{P}$, so we may assume that $S_{j-1} = S'_j$, for $j = 3, \ldots, k-1$. Consider now a configuration X in the support of w. We will show that there is a configuration Y in the support of w' such that $|Y - X| \le 1$. Let x_k be the last page of X in the canonical ordering. Also, let b be the first page of the canonical ordering not in $X - x_k$. We claim that $Y = X - x_k + b$, which differs in at most one position from X, is in the support of w'. Notice first that Y contains the page in S_1. It also contains the second page of the canonical ordering because either this page is in $X - x_k$ or it is equal to b. It remains to show that $|Y \cap S'_j| \ge j$, for $j = 3, \ldots, k$. There are two cases to consider: The first case, when $|X \cap S_{j-1}| \ge j$, follows from the fact that x_k is in S'_j $(= S_{j-1})$ only if b is in S'_j. For the second case, when $|X \cap S_{j-1}| = j - 1$, it suffices to note that $b \in S'_j$ but $x_k \notin S'_j$. □

2.2. Optimality of LRU. We now turn to the Δ_ϵ diffuse adversary. Our goal in this section is to show that LRU has optimal competitive ratio against a diffuse adversary. The usual approach to show that an on-line algorithm has optimal competitive ratio is to compute its ratio and then show that every other algorithm has no smaller competitive ratio. The difficulty here is that we cannot compute the competitive ratio of LRU. Thus we have to compare directly LRU with any other on-line algorithm. More precisely, we have to show that for any on-line algorithm A and any adversary (conditional probability distribution) $D \in \Delta_\epsilon$, there is an adversary $D' \in \Delta_\epsilon$ such that the competitive ratio of A against D' is at least the competitive ratio of LRU against D. We have to face the problem that D may be very complicated and A may be erratic.

We attack the problem in two steps. First, we "standardize" the class of adversaries, by showing that it suffices to consider only a class of adversaries which we call *conservative* (to be defined shortly). In particular, we will show that for any $D \in \Delta_\epsilon$ there is a better conservative adversary $\hat{D} \in \Delta_\epsilon$ against LRU, i.e., LRU has a competitive ratio against $\hat{D}$ which is no less than that against D. Second, we show that for any conservative $D \in \Delta_\epsilon$ against LRU, there is a conservative adversary $D' \in \Delta_\epsilon$ against A, such that the competitive ratio of LRU against D is at most the competitive ratio of A against D'.

To understand why LRU is optimal against diffuse adversaries and to motivate the notion of conservatives adversaries, we start by asking what property an optimal on-line algorithm must have. Intuitively, if $(S_1, \ldots, S_k)$ is a signature of the current work function w, an optimal on-line algorithm should prefer to have in its fast memory pages from S_i instead of pages not in S_i. The intuition is that pages in S_i are more valuable to the off-line algorithm than pages not in S_i, because a configuration from the support of w remains in the support when we replace any page not in S_i with a page in S_i; the converse does not hold in general. LRU does exactly this. In fact, LRU keeps in its fast memory the first k pages of the canonical ordering (with appropriate initialization).

The same intuition suggests that the adversary should prefer as next request a page $a \in S_i$ to a page $b \notin S_i$; the only exception being when a is in the on-line cache and b is not. In this case, it is unclear which page (a or b) the adversary should choose because of the following tradeoff: On the one hand, b is more favorable because it increases the on-line cost, whereas a does not. On the other hand, a is more favorable since either the resulting support is larger or b increases the off-line cost. The above intuition suggests the notion of conservative adversary. A conservative adversary assigns probability that favors pages with smaller rank in the canonical ordering among the pages in the on-line cache and similarly for the pages not in the on-line cache. To be more precise, fix a request sequence ρ that results in canonical ordering $a_1 a_2 \ldots$ and configuration C of an on-line algorithm A. A conservative adversary against A assigns probabilities with the property that for every $a_i, a_j \in C$ with $i < j$, a_j receives positive probability, $D(a_j|\rho) > 0$, only if a_i receives maximum probability, $D(a_i|\rho) = \epsilon$. Similarly for pages $a_i, a_j \notin C$. Thus the probabilities $D(\cdot|\rho)$ are completely determined by the total probability z assigned to pages in C. We add an additional constraint, although the proofs do not call for it: At most one page receives probability that is not 0 or ϵ (equivalently either z or $1 - z$ is an integral multiple of ϵ). For example, if $\epsilon = 1/5$, the on-line cache C consists of pages $1, 3, 4, 5$ of the canonical ordering, and $z = 2/5$, then pages $1, 2, 3, 6, 7$ receive probability ϵ.

It seems reasonable that an optimal adversary against LRU is conservative and indeed we are going shortly to prove that this is the case. For other on-line algorithms however, conservative adversaries may not be optimal, especially for "unreasonable" and highly suboptimal on-line algorithms. A central idea of our proof is to disregard this problem. Even if conservative adversaries are not optimal against an on-line algorithm A, it suffices to show that there is a conservative adversary that forces a competitive ratio no less than the ratio of LRU. We now show that an optimal adversary against LRU is conservative.

LEMMA 2.6. *For any $D \in \Delta_\epsilon$, there is a conservative adversary $\hat{D} \in \Delta_\epsilon$ such that the competitive ratio of LRU against $\hat{D}$ is at least the competitive ratio of LRU against D.*

Proof. The proof is by induction on the number of requests. To facilitate induction, we use a strong inductive hypothesis. First, we allow any initial work function (instead of the work function whose support contains only one configuration). Let

$\mathrm{opt}(\rho, \mathbb{P}, C)$ be the optimal cost to service ρ when the initial work function has type $\mathbb{P}$ and canonical ordering C. For simplicity, in our notation we use only the type of work functions and we write $\mathrm{opt}(\rho, \mathbb{P})$ instead of $\mathrm{opt}(\rho, \mathbb{P}, C)$. Notice also that the initial type does not affect the behaviour of LRU at all, so we can simply write $\mathrm{LRU}(\rho)$ to denote the cost of *LRU* for the sequence request ρ.

Second, since we don't know the competitive ratio of LRU, we simply use the fact that the competitive ratio is positive (although we could safely assumed a competitive ratio at least 1). More precisely, the inductive hypothesis is that for any $D \in \Delta_\epsilon$, any n, and any $c > 0$ there is a conservative $\hat{D}$ such that $\mathcal{E}_D(LRU(x_n) - c \cdot \mathrm{opt}(x_n, \mathbb{P})) \leq \mathcal{E}_{\hat{D}}(LRU(x_n) - c \cdot \mathrm{opt}(x_n, \mathbb{P}))$, where $\mathbb{P}$ is the type of the initial work function, x_n denotes a sequence of length n drawn from the conditional probability distribution D or $\hat{D}$. Equivalently, we will show that there is a conservative D that maximizes

$$\psi(D, n, \mathbb{P}) = \mathcal{E}_D(LRU(x_n) - c \cdot \mathrm{opt}(x_n, \mathbb{P})). \tag{2.2}$$

We use induction on the number of requests n. For $n = 0$, there is nothing to prove. Assume now that the induction hypothesis holds for $n-1$ and let D be a distribution that maximizes $\mathcal{E}_D(LRU(x_n) - c \cdot \mathrm{opt}(x_n, \mathbb{P}))$. If D is not conservative, we will show how to alter it to get a conservative adversary $\hat{D}$ that also maximizes $\psi(\hat{D}, n, \mathbb{P})$.

The proof proceeds as follows: Although D may not be conservative, by induction it becomes conservative after the first request. So, we want to show that the first request is also "conservative". Denote by D_j the resulting conditional distribution when the first request is the j-th request a_j of the canonical ordering, i.e., $D_j(\cdot|\rho) = D(\cdot|a_j\rho)$. Let also $\mathbb{P}^{(j)}$ denote the type of the work function that results after request a_j. If $(S_1, \ldots, S_k)$ is the initial canonical signature, then

$$\psi(D, n, \mathbb{P}) = \sum_{j \geq 1} D(a_j) \cdot \left[\psi(D_j, n-1, \mathbb{P}^{(j)}) + I(j > k) - c \cdot I(a_j \notin S_k)\right], \tag{2.3}$$

where $D(a_j) = D(a_j|\varepsilon)$ is the probability assigned to page a_j conditioned on the empty sequence ε, and $I(\phi)$ is the indicator function that takes value 1 when ϕ is true and 0 otherwise. The last two terms inside the sum of the right-hand side correspond to the on-line and the off-line cost for the first request. By induction, we can replace D_j with a conservative $\hat{D}_j$ without decreasing the right-hand side of (2.3). So, without loss of generality, we assume that the distributions D_j are conservative.

If we fix the conditional distributions D_j, the optimal probabilities for the first request can be computed as follows: order the pages in decreasing $\psi(D_j, n-1, \mathbb{P}^{(j)}) + I(j > k) - c \cdot I(a_j \notin S_k)$ value, assign probability ϵ to the first $\lfloor 1/\epsilon \rfloor$ pages and assign the remaining probability (which of course is less than ϵ) to the next page.

We simply want to guarantee that the probabilities are assigned in a conservative manner. Equivalently, if $i < j \leq k$ (both a_i and a_j are in LRU's cache) or when $k < i < j$ (neither a_i nor a_j are in LRU's cache), it suffices to show that

$$\begin{aligned} &\psi(D_i, n-1, \mathbb{P}^{(i)}) + I(i > k) - c \cdot I(a_i \notin S_k) \geq \\ &\psi(D_j, n-1, \mathbb{P}^{(j)}) + I(j > k) - c \cdot I(a_j \notin S_k). \end{aligned} \tag{2.4}$$

We first consider the case of $i < j \leq k$. The crucial point for establishing (2.4) is that we can assume:

$$\psi(D_i, n-1, \mathbb{P}^{(i)}) \geq \psi(D_j, n-1, \mathbb{P}^{(j)}). \tag{2.5}$$

To see this, let us consider the adversary $\hat{D}$ that is identical to D but when the first request is a_i, it continues like D_j instead of D_i (taking into account the difference in the canonical orderings). Formally, if $b_1 b_2 \cdots$ is the canonical ordering that results when the first request is a_j and $b'_1 b'_2 \cdots$ when the first request is a_i, then $\hat{D}(b'_l | a_i b'_{l_1} \cdots b'_{l_m}) = D(b_l | a_j b_{l_1} \cdots b_{l_m})$. We then have $\psi(\hat{D}, n, \mathbb{P}) = \psi(D, n, \mathbb{P}) + D(a_i)(\psi(D_j, n-1, \mathbb{P}^{(i)}) - \psi(D_i, n-1, \mathbb{P}^{(i)}))$.

But since $\mathbb{P}^{(i)} \geq \mathbb{P}^{(j)}$, we can apply Lemma 2.5 and get $\text{opt}(x_{n-1}, \mathbb{P}^{(i)}) \leq \text{opt}(x_{n-1}, \mathbb{P}^{(j)})$ which in turn implies that $\psi(D_j, n-1, \mathbb{P}^{(i)}) \geq \psi(D_j, n-1, \mathbb{P}^{(j)})$. Thus $\psi(\hat{D}, n, \mathbb{P}) \geq \psi(D, n, \mathbb{P}) + D(a_i)(\psi(D_j, n-1, \mathbb{P}^{(j)}) - \psi(D_i, n-1, \mathbb{P}^{(i)}))$. If (2.5) does not hold, then $\psi(\hat{D}, n, \mathbb{P}) \geq \psi(D, n, \mathbb{P})$; hence, $\hat{D}$ maximizes ψ and of course has the desired property (2.5).

Combining (2.5) and the fact that $I(i > k) = I(j > k) = I(a_i \notin S_k) = I(a_j \notin S_k) = 0$, we get that (2.4) holds for $i < j \leq k$.

Inequality (2.4) holds for $k < i < j$. An identical argument as above establishes it for the case of $a_i, a_j \in S_k$ or $a_i, a_j \notin S_k$. The remaining case, when $a_i \in S_k$ and $a_j \notin S_k$, is treated similarly. In this case, we have that $\mathbb{P}^{(i)} \trianglerighteq \mathbb{P}^{(j)}$, which implies $\text{opt}(x_{n-1}, \mathbb{P}^{(i)}) \leq \text{opt}(x_{n-1}, \mathbb{P}^{(j)}) + 1$. The last inequality implies $\psi(D_i, n-1, \mathbb{P}^{(i)}) \geq \psi(D_j, n-1, \mathbb{P}^{(j)}) - c$, and together with the fact that $I(i > k) = I(j > k) = 1$, $I(a_i \notin S_k) = 0$, and $I(a_j \notin S_k) = 1$, we get (2.4). □

The above lemma establishes that the best adversary against LRU is conservative. We can now proceed to the second part of the proof that LRU has optimal competitive ratio. We will show that for any on-line algorithm A and any conservative adversary $D \in \Delta_\epsilon$ against LRU, there is an adversary $D' \in \Delta_\epsilon$ against A, such that the competitive ratio of LRU against D is at most the competitive ratio of A against D'. The main idea for constructing a D' is by enforcing the on-line cost of LRU against D to be equal to the on-line cost of A against D'. This allows us to compare the competitive ratios of LRU and A, by simply comparing the corresponding off-line costs.

Given D, how can we design D' so that cost of LRU against D is equal to the cost of A against D'? For the first request it is obvious: the probability that D' assigns to pages in the cache of A should be equal to the probability that D assigns to pages in LRU's cache. But for the second and subsequent requests, the situation depends on previous requests (the outcome of random experiments). In general, a conservative adversary against an algorithm B corresponds to an (infinite) rooted tree T with outdegree $\lceil 1/\epsilon \rceil$ such that each node v has weight $z(v)$ in $[0, \min\{k\epsilon, 1\}]$; furthermore, either $z(v)$ or $1 - z(v)$ is an integral multiple of ϵ. Paths on this tree correspond to request sequences and the values $z(v)$ are the total probability assigned by the adversary to pages in the on-line cache. More precisely, a request sequence is produced by starting at the root and descending down the tree. At a node v the adversary assigns probabilities in a conservative manner so that the total probability assigned to pages of the current cache of B is $z(v)$. Since the adversary is conservative, at most $l = \lceil 1/\epsilon \rceil$ pages receive non-zero probability. Let $r_1, \ldots, r_l$ be these pages (first the pages in the current cache of B and then the pages outside the cache in the canonical order). If the next request (the outcome of the random experiment) is r_j, the adversary moves to the j-th child of v and repeats the process to produce the next request. We will call this conservative adversary the *adversary based on T against B* and we will denote it by $D_{T(B)}$. Notice that the conditional probability distribution $D_{T(B)}$ depends on the on-line algorithm B (for another algorithm B' it may be different). It also depends on the initial configuration of B, although for simplicity we omit this dependance in our notation.

It is now simple to design D' so that the expected cost of A against D' is equal to the expected cost of LRU against D: if T is the tree such that $D = D_{T(\mathrm{LRU})}$ then $D' = D_{T(A)}$. It suffices to show that the expected optimum cost against D is no less than the expected optimum cost against D'. More precisely, we will show that the expected off-line cost to service a request sequence of length n produced by $D_{T(\mathrm{LRU})}$ is no less than the off-line cost to service a request sequence of length n produced by $D_{T(A)}$.

We will use induction on the length n of the request sequence. In order to facilitate induction, we generalize the problem by assuming any initial work function. As in the proof of Lemma 2.6, our notation will include only the type of the work function. More precisely, let $h_{T(A)}(n, \mathbb{P})$ denote the expected off-line cost to service a request sequence of length n produced by the conditional distribution $D_{T(A)}$ when the initial work function has type $\mathbb{P}$. We will show that for every $\mathbb{P}$: $h_{T(\mathrm{LRU})}(n, \mathbb{P}) \geq h_{T(A)}(n, \mathbb{P})$. But first, we need the following simple lemma.

LEMMA 2.7. *For all conservative adversaries T, if $\mathbb{P} \leq \mathbb{Q}$ then $h_{T(LRU)}(n, \mathbb{P}) \geq h_{T(LRU)}(n, \mathbb{Q})$, and if $\mathbb{P} \trianglelefteq \mathbb{Q}$ then $h_{T(LRU)}(n, \mathbb{P}) + 1 \geq h_{T(LRU)}(n, \mathbb{Q})$.*

Proof. The crucial observation is that LRU does not depend on the initial type of the work function, but only on the initial canonical ordering. An immediate consequence is that the conditional probability distribution $T(\mathrm{LRU})$ is independent of the initial work function. Consider first the case of $\mathbb{P} \leq \mathbb{Q}$. By Lemma 2.5, the off-line cost to service a request sequence starting with a work function w of type $\mathbb{P}$ cannot be less that the off-line cost to service the same request sequence starting with a work function w' of type $\mathbb{Q}$ and the same canonical ordering with w. Therefore, $h_{T(\mathrm{LRU})}(n, \mathbb{P}) \geq h_{T(\mathrm{LRU})}(n, \mathbb{Q})$. The other case, $\mathbb{P} \trianglelefteq \mathbb{Q}$, is handled similarly. $\square$

We are now ready to show that the expected off-line cost of a request sequence produced by a tree T is maximized when the on-line algorithm is the LRU algorithm.

LEMMA 2.8. *For every tree T and every on-line algorithms A, $h_{T(LRU)}(n, \mathbb{P}) \geq h_{T(A)}(n, \mathbb{P})$.*

Proof. By induction on n. For $n = 0$, the lemma is trivially true. Assume that the lemma holds for $n - 1$. Denote by T_j the subtree rooted at the j-th child of the root of T. Let r_A and r_{LRU} be the request that produced when the adversary descends to T_j for algorithms A and LRU, respectively. Let also $\mathbb{P}_{i_A}$, $\mathbb{P}_{i_{\mathrm{LRU}}}$ denote the resulting type after requests r_A and r_{LRU}. It is important to notice that $i_A \leq i_{\mathrm{LRU}}$; this is the only property of algorithm A used in the proof.

Child j is chosen with probability that depends on the value $z(v)$, where v is the root of T. This probability is ϵ for all values of j except of one child when $1/\epsilon$ is not an integer. Obviously, $h_{T(A)}(n, \mathbb{P})$ is equal to the expected value of $h_{T_j(A)}(n-1, \mathbb{P}_{i_A})$ plus the off-line cost for the first request; the off-line cost is 0 when $i_A < k$ and it is 1 when $i_A = k$. In other words $h_{T(A)}(n, \mathbb{P}) = E[h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) + I(i_A = k)]$. A similar expression holds also for $h_{T(LRU)}(n, \mathbb{P})$. It suffices therefore to show that for all j: $h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) + I(i_A = k) \leq h_{T_j(LRU)}(n-1, \mathbb{P}_{i_{\mathrm{LRU}}}) + I(i_{\mathrm{LRU}} = k)$.

We consider three cases according to the values of i_A and i_{LRU}. In the first case, $i_A \leq i_{\mathrm{LRU}} < k$, we get

$$\begin{aligned} h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) + I(i_A = k) &= h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) \\ &\leq h_{T_j(\mathrm{LRU})}(n-1, \mathbb{P}_{i_A}) \\ &\leq h_{T_j(\mathrm{LRU})}(n-1, \mathbb{P}_{i_{\mathrm{LRU}}}) \\ &= h_{T_j(\mathrm{LRU})}(n-1, \mathbb{P}_{i_{\mathrm{LRU}}}) + I(i_{\mathrm{LRU}} = k), \end{aligned}$$

where the first inequality follows from the induction hypothesis and the second one from Lemma 2.7, because $\mathbb{P}_{i_A} \geq \mathbb{P}_{i_{\rm LRU}}$. In the second case, $i_A < i_{\rm LRU} = k$, we get $h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) + I(i_A = k) = h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) \leq h_{T_j({\rm LRU})}(n-1, \mathbb{P}_{i_A}) \leq h_{T_j({\rm LRU})}(n-1, \mathbb{P}_{i_{\rm LRU}}) + 1 = h_{T_j({\rm LRU})}(n-1, \mathbb{P}_{i_{\rm LRU}}) + I(i_{\rm LRU} = k)$. Again, the first inequality follows from the induction hypothesis and the second one from Lemma 2.7, because $\mathbb{P}_{i_A} \trianglerighteq \mathbb{P}_{i_{\rm LRU}}$. Finally, the third case, $i_A = i_{\rm LRU} = k$, is handled similarly: $h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) + I(i_A = k) = h_{T_j(A)}(n-1, \mathbb{P}_{i_A}) + 1 \leq h_{T_j({\rm LRU})}(n-1, \mathbb{P}_{i_{\rm LRU}}) + 1 = h_{T_j({\rm LRU})}(n-1, \mathbb{P}_{i_{\rm LRU}}) + I(i_{\rm LRU} = k)$. $\square$

The above lemmata establish the main result of this section.

THEOREM 2.9. *For any ϵ, LRU has optimal competitive ratio $R(\Delta_\epsilon)$ for the paging diffuse adversary model.*

The above lemmata however do not provide any efficient way to estimate the competitive ratio $R(\Delta_\epsilon)$. The approach suggested by the lemmata is to nondeterministically guess the optimal conservative adversary and then compute the competitive ratio of LRU against this adversary. A conservative adversary is determined by the values $z(v)$ that are multiples of ϵ when $1/\epsilon$ is an integer; it is slightly more complicated when $1/\epsilon$ is not an integer. For a given conservative adversary, the competitive ratio of LRU is given by a finite Markov chain; the states of the Markov chain are all reachable types and there are at most $(k+1/\epsilon)^{k-1}$ such types. This approach provides a doubly exponential algorithm (approximately $k^{(k+1/\epsilon)^{k-1}}$) to compute $R(\Delta_\epsilon)$. It is an interesting open problem to determine $R(\Delta_\epsilon)$, as a function of ϵ. For the extreme values of ϵ, we know that $R(\Delta_1) = k$ and $\lim_{\epsilon\to 0} R(\Delta_\epsilon) = 1$. In the first case, the adversary has complete power and in the second case, it suffers a page fault in almost every step. Recently, Young [16] estimated $R(\Delta_\epsilon)$ within (almost) a factor of two; $R(\Delta_\epsilon)$ is between $\Phi(\epsilon,k)-1$ and $2\Phi(\epsilon,k)$ where $\Phi(\epsilon,k) = 1+\sum_{i=1}^{k-1} 1/\max\{1/\epsilon - i, 1\}$ (this is approximately $\ln \frac{1}{1-(k-1)\epsilon}$ for $\epsilon > 1/k$, and $k+1-1/\epsilon$ otherwise).

Even for the simplest case of $k=2$, determining the competitive ratio is not trivial. We give here a lower bound which we believe is exact. This is the competitive ratio against the conservative adversary that always assigns non-zero probability to exactly one page from the pages in the on-line fast memory.

PROPOSITION 2.10. *For $k=2$, if $n = 1/\epsilon - 1$ is an integer then*

$$R(\Delta_\epsilon) \geq \frac{\sum_{i=0}^{n} n^i/i!}{\sum_{i=0}^{n-1} n^i/i!} \in [1+\sqrt{\epsilon}/2, 1+2\sqrt{\epsilon}].$$

Proof. Consider the conservative adversary that always assigns probability ϵ to pages of the current canonical ordering with rank $1, 3, 4, \ldots, 1/\epsilon+1$. Notice that page 2 is assigned zero probability. The request on the page of rank 1 is identical to the previous request and does not change anything. Thus, this adversary is equivalent to the adversary that assigns probability $\delta = \epsilon/(1-\epsilon) = 1/n$ to pages with rank $3, 4, \ldots, 1/\epsilon+1$.

Let $(1, p_2)$ be the type of the current work function. The following table summarizes the possibilities of the next request, together with the probability, the resulting type, and the associated on-line and off-line cost for servicing the request.

Request	Probability	Type	On-line cost	Off-line cost
$3 \ldots p_2$	$(p_2-2)\delta$	$(1,2)$	1	0
$p_2+1 \ldots 1/\delta+2$	$1-(p_2-2)\delta$	$(1,p_2+1)$	1	1

In summary, the competitive ratio is given by a Markov chain M_δ with states the types $(1, p_2)$ for $p_2 = 2, 3, \ldots, 1/\delta+2$. The transition probabilities from state $(1, p_2)$

are given in the above table. It is not difficult to see that the Markov process is identical to the following random process: In each phase repeatedly choose uniformly a number from $\{1, 2, \ldots, n\}$, where $n = 1/\delta$; a phase ends when a number is chosen twice. The state $(1, p_2)$ of M_δ corresponds to the case that $p_2 - 2$ numbers have been drawn. This random process is a generalization of the well-known *birthday* problem in probability theory. A phase corresponds to a cycle in the Markov chain that starts (and ends) at state with type $(1, 2)$. The expected off-line cost per phase is equal to the length of a phase minus one (all transitions in the cycle have off-line cost one except the last one). Similarly, the expected on-line cost per phase is equal to the length of a phase (all transitions have on-line cost one). It is not hard now to verify the expression for $R(\Delta_\epsilon)$.

For the purpose of bounding the expected length of a phase, notice that each of the first $\sqrt{n}$ numbers has probability at most $1/\sqrt{n}$ to end the phase. In contrast, each of the next $\sqrt{n}$ numbers has probability at least $1/\sqrt{n}$ to end the phase. Elaborating on this observation we get that R_ϵ is in the interval $[1 + \sqrt{\epsilon}/2, 1 + 2\sqrt{\epsilon}]$.

Numerical evaluations suggest that the value of R_ϵ is approximately $1 + 0.8\sqrt{\epsilon}$, when $\epsilon \to 0$. □

A preliminary version of this work [8] had an incorrect proof of optimality of LRU. The proof was based on the unjustified assumption that a conservative adversary that achieves optimal competitive ratio against LRU assigns probability ϵ to exactly one page in the on-line fast memory. As was pointed out to us by Neal Young (see also [16]), this assumption does not hold in general.

An important open problem is to determine the competitive ratio of known paging algorithms against a diffuse adversary. The most important direction is to estimate the competitive ratio of FIFO. The recent work of Young [16] estimates the competitive ratio of marking algorithms —both LRU and FIFO are marking algorithms— almost within a factor of 2. He gives similar bounds for randomized algorithms. Our proof of the optimality of LRU seems to suggest that for certain values of ϵ, the competitive ratio of FIFO is not optimal. In particular, FIFO does not always keep in its fast memory the first k pages of the canonical ordering or an equivalent set of pages. This then is an indication that a conservative adversary may force FIFO to have larger cost than LRU. We conjecture that FIFO is suboptimal for some values of ϵ. If indeed this is the case, it will add some extra validity to the paging diffuse adversary model, in the sense that the model can actually distinguish between LRU and FIFO.

3. Comparative Analysis. On-line algorithms deal with the relations between *information regimes*. Formally but briefly, an information regime is the class of all functions from a domain D to a range R which are constant within a fixed partition of D. Refining this partition results in a richer regime. Traditionally, the literature on on-line algorithms has been preoccupied with comparisons between two basic information regimes: The *on-line* and the *off-line* regime (the off-line regime corresponds to the fully refined partition). As we argued in the introduction, this has left unexplored several intricate comparisons between other important information regimes.

Comparative analysis is a generalization of competitive analysis allowing comparisons between arbitrary information regimes, via the comparative ratio defined in equation (1.3). Naturally, such comparisons make sense only if the corresponding regimes are rich in algorithms—single algorithms do not lend themselves to useful comparisons. As in the case of the competitive ratio for the diffuse adversary model, we usually allow an additive constant in the numerator of equation (1.3).

We apply comparative analysis in order to evaluate the power of *lookahead* in task systems. An on-line algorithm for a metrical task system has lookahead ℓ if it can base its decision not only on the past, but also on the next ℓ requests. All on-line algorithms with lookahead ℓ comprise the information regime $\mathcal{L}_\ell$. Thus, $\mathcal{L}_0$ is the class of all traditional on-line algorithms.

Metrical task systems [3] are defined on some metric space $\mathcal{M}$; a server resides on some point of the metric space and can move from point to point. Its goal is to process on-line a sequence of tasks $T_1, T_2, \ldots$ The server is free to move to any position before processimg a task, although it has to pay the distance. The cost $c(T_j, a_j)$ for processing a task T_j is determined by the task T_j and the position a_j of the server while processing the task. The total cost for processing the sequence is the sum of the distance moved by the server plus the cost of servicing each task T_j, $j = 1, 2, \ldots$

THEOREM 3.1. *For any metrical task system, $R(\mathcal{L}_0, \mathcal{L}_\ell) \leq 2\ell + 1$. Furthermore, there are metrical task systems for which $R(\mathcal{L}_0, \mathcal{L}_\ell) = 2\ell + 1$.*

Proof. Trivially the theorem holds for $\ell = 0$. Assume that $\ell > 0$ and fix an algorithm B in $\mathcal{L}_\ell$. We shall define an on-line algorithm A without lookahead whose cost on any sequence of tasks is at most $2\ell + 1$ times the cost of B. Algorithm A is a typical on-line algorithm in comparative analysis: It tries to efficiently "simulate" the more powerful algorithm B. In particular, A knows the position of B ℓ steps ago. In order to process the next task, A moves first to B's last known position, and then processes the task greedily, that is, with the minimum possible cost.

Let $T_1, T_2, \ldots$ be a sequence of tasks and let $b_1, b_2, \ldots$ be the points where algorithm B processes each task and $a_1, a_2, \ldots$ the corresponding points for algorithm A. For simplicity, we define also points $b_j = a_j = a_0$ for negative j's.

Then the cost of algorithm B is

$$\sum_{j \geq 1} (d(b_{j-1}, b_j) + c(T_j, b_j))$$

and the cost of algorithm A is

$$\sum_{j \geq 1} (d(a_{j-1}, b_{j-\ell}) + d(b_{j-\ell}, a_j) + c(T_j, a_j)) . \tag{3.1}$$

Recall that in order to process the j-th task, algorithm A moves to B's last known position $b_{j-\ell}$ and then processes the task greedily, that is, $d(b_{j-\ell}, a_j) + c(T_j, a_j)$ is the smallest possible. In particular,

$$d(b_{j-\ell}, a_j) + c(T_j, a_j) \leq d(b_{j-\ell}, b_j) + c(T_j, b_j).$$

From this, the fact that costs are nonnegative, and the triangle inequality we get

$$\begin{aligned} d(a_{j-1}, b_{j-\ell}) &\leq d(a_{j-1}, b_{j-\ell-1}) + d(b_{j-\ell-1}, b_{j-\ell}) \\ &\leq d(b_{j-1}, b_{j-\ell-1}) + c(T_{j-1}, b_{j-1}) + d(b_{j-\ell-1}, b_{j-\ell}) \end{aligned}$$

We can now bound the cost of algorithm A in (3.1) by

$$\sum_{j \geq 1} (d(b_{j-1}, b_{j-\ell-1}) + c(T_{j-1}, b_{j-1}) + d(b_{j-\ell-1}, b_{j-\ell}) + d(b_{j-\ell}, b_j) + c(T_j, b_j)) \tag{3.2}$$

Using the triangle inequalities of the form

$$d(b_i, b_{i+2}) \leq d(b_i, b_{i+1}) + d(b_{i+1}, b_{i+2})$$

we can expand $d(b_{j-\ell-1}, b_{j-1}) \leq d(b_{j-\ell-1}, b_{j-\ell}) + \cdots + d(b_{j-2}, b_{j-1})$ and similarly we can expand $d(b_{j-\ell}, b_j)$. Observe now that each term $d(b_{i-1}, b_i)$ appears in (3.2) $2\ell+1$ times and each term $c(T_i, b_i)$ appears twice. We can therefore conclude that the cost of algorithm A is at most

$$\sum_{j\geq 1} ((2\ell+1)d(b_{j-1}, b_j) + 2c(T_j, b_j)) \leq (2\ell+1) \sum_{j\geq 1} (d(b_{j-1}, b_j) + c(T_j, b_j))$$

The last expression is $(2\ell+1)$ times the cost of algorithm B.

To show the converse, we consider a task system with metric space $\mathcal{M}$ a (rooted) binary tree, where the distance between adjacent vertices is 1. Let B be the "greedy" algorithm with lookahead ℓ. In other words, B services the next task in such a way that minimizes the total cost to service the next ℓ tasks. Consider now an algorithm A with no lookahead. We will describe a sequence of tasks $T_1, T_2, \ldots$ that force a comparative ratio $2\ell+1$ for A against B. For this, let a_{j-1} be the position where A services the task T_{j-1}. With appropriate initialization, assume that a_{j-1} is at depth $j+\ell$. The next task T_j has infinite cost on all vertices except for the 2^ℓ vertices that are on depth $j+\ell+1$ and on distance $2\ell+1$ from the current position a_{j-1} (to move to one of these vertices A must move up to level j and then down to level $j+\ell+1$). The cost of T_j on these vertices is 0. Thus, the cost for A to service each such task is $2\ell+1$, while the cost for B is 1 (using its lookahead power, it simply walks down the tree).

We remark here that although the above lower bound uses an infinite metric space, a finite metric space that looks locally like a binary tree can also be used. In particular, consider a butterfly (the FFT graph) of $2\ell+2$ levels and identify the first and last level. Then we can embed the infinite binary tree into this graph in such a way that every subtree of $\ell+1$ levels is embedded isometrically (in a distance-preserving manner). We conclude that there are task systems with metric spaces of $2^{O(\ell)}$ points and comparative ratio $2\ell+1$. We leave it as an interesting open problem to determine the comparative ratio for smaller metric spaces. $\square$

Of course, for certain task systems the comparative ratio may be less that $2\ell+1$. For the paging problem, it is $\min\{\ell+1, k\}$.

THEOREM 3.2. *For the paging problem*

$$R(\mathcal{L}_0, \mathcal{L}_\ell) = \min\{\ell+1, k\}.$$

Proof. Let $n = \min\{\ell, k-1\}$ and let B be an algorithm for the paging problem in the class $\mathcal{L}_\ell$, that is, with lookahead ℓ. Without loss of generality we assume that B moves pages only to service requests. Consider the following on-line algorithm A which is a generalization of LRU:

> To service a request r not in its fast memory, A evicts a page that is not one of the n most recent distinct requests (including r). Among the remaining pages, A chooses to evict a page such that the resulting configuration is as close as possible to the last known configuration of B. A does nothing for requests in its fast memory.

To show that the comparative ratio of A is $n+1$, it suffices to show that for every $n+1$ consecutive page faults of A, B suffers at least one page fault. This can be achieved by showing that after each subsequence of requests that causes n page faults to A and no page fault to B, both algorithms are in the same configuration. To do this, we show by induction on the number of requests the stronger claim that after a subsequence of requests that cause c page faults to A and no fault to B, the configurations of A and B differ by at most $n-c$ pages.

Fix a request sequence $\rho = r_1 r_2 \dots$ and let $A_0, A_1, \dots$ and $B_0 = A_0, B_1, \dots$ be the configurations of A and B that service ρ. The base of the induction is trivial. Assume that the induction hypothesis holds for $t-1$. We have to deal with a few cases. First of all, when $r_t \in A_{t-1}$, A suffers no page fault, and the inductive step follows from the fact that $|A_t - B_t| \leq |A_{t-1} - B_{t-1}|$. 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 $r_t \in B_{t-1} - A_{t-1}$. 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 $A_{t-1} \subseteq B_{t-n} + \{r_{t-n+1}, \dots, r_{t-1}\}$ (otherwise A would not choose $x_t \in B_{t-n}$ to evict) and

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

We also have the obvious relation

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

(recall that we assumed that B moves pages only to service requests). If B suffered no page fault during the last c requests, i.e., $B_t = B_{t-1} = \cdots B_{t-c}$, the set $B_{t-n} + \{r_{t-n+1}, \dots, r_{t-1}, r_t\}$ 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 a lower bound of the comparative ratio, let B be a variant of the optimal off-line algorithm adapted to lookahead ℓ. More precisely, B never evicts one of the next n requests. Fix a set of $k+1$ pages. Clearly, 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. Open problems.

We introduced two refinements of competitive analysis, the diffuse adversary model and comparative analysis. Both restrict the power of the adversary: the first by allowing only certain input distributions and the second by restricting the refinement of the adversary's information regime. In general, we believe that the only natural way to deal with uncertainty is by designing algorithms that perform well in the worst world which is compatible with the algorithm's knowledge.

There are a lot of interesting open problems suggested by this approach. The most important open problem is to determine the competitive ratio of FIFO for the paging diffuse adversary model. As mentioned above, we conjecture that FIFO is in general suboptimal. There are numerous other applications of these two frameworks for evaluating on-line algorithms. We simply mention here two challenging open problems.

The Markov diffuse adversary. Consider again the paging problem. Suppose that the request sequence is the output sequence of an unknown *Markov chain* (intuitively, the program generating the page requests) with at most s states, which we can only partially observe via its output. That is, the output $f(q)$ of a state q of the unknown Markov process is a page in M. The allowed distributions Δ are now all output distributions of s-state Markov processes with output. We may want to restrict our on-line algorithms to ones that do not attempt to exhaustively learn the Markov process. One way to do this would be to bound the length of the request sequence to $O(s)$. A better way, however, is to require the additive constant d in (2.1) to be independent of s. We believe that this is a useful model of paging, whose study and solution may enhance our understanding of the performance of actual paging systems.

The power of vision. Consider two robots, one with vision α (its visual sensors can detect objects in distance α) and the other with vision β, $\beta > \alpha$. We want to measure the disadvantage of the first robot in navigating or exploring a terrain against the second robot. Naturally, comparative analysis seems the most appropriate framework for this type of problems. Different restrictions on the terrain and the objective of the robot result in different problems but we find the following simple problem particularly challenging: On the plane, there are n objects (points). The objective of the robot is to construct a map of the plane, i.e., to find the position of all n objects. We ask what the comparative ratio $R(\mathcal{V}_\alpha, \mathcal{V}_\beta)$ for this problem is, where $\mathcal{V}_\alpha$ and $\mathcal{V}_\beta$ denote the information regimes of vision α and β, respectively.

REFERENCES

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

[2] A. Borodin, S. Irani, P. Raghavan, and B. Schieber, *Competitive paging with locality of reference.*, Proceedings 23rd Annual ACM Symposium on Theory of Computing, (1991), pp. 249–59.

[3] A. Borodin, N. Linial, and M. E. Saks, *An optimal online algorithm for metrical task systems.*, Proceedings 19th Annual ACM Symposium on Theory of Computing, (1987), pp. 373–82.

[4] M. Chrobak and L. L. Larmore, *The server problem and on-line games.*, On-line algorithms: proceedings of a DIMACS workshop. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 7 (1992), pp. 11–64.

[5] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young, *Competitive paging algorithms.*, Journal of Algorithms, 12 (1991), pp. 685–99.

[6] S. Irani, A. R. Karlin, and S. Phillips, *Strongly competitive algorithms for paging with locality of reference.*, Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms, (1992), pp. 228–36.

[7] A. R. Karlin, S. J. Phillips, and P. Raghavan, *Markov paging.*, Proceedings 33rd Annual Symposium on Foundations of Computer Science, (1992), pp. 208–17.

[8] E. Koutsoupias and C. H. Papadimitriou, *Beyond competitive analysis*, in Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 394–400.

[9] ———, *On the k-server conjecture.*, Proceedings 26th Annual ACM Symposium on Theory of Computing, (1994), pp. 507–11.

[10] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, *Competitive algorithms for on-line problems.*, Proceedings 20th Annual ACM Symposium on Theory of Computing, (1988), pp. 322–33.

[11] L. A. McGeoch and D. D. Sleator, *A strongly competitive randomized paging algorithm.*, Algorithmica, 6 (1991), pp. 816–25.

[12] P. Raghavan, *A statistical adversary for on-line algorithms.*, On-line algorithms: proceedings of a DIMACS workshop. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 7 (1992), pp. 79–83.

[13] P. Raghavan and M. Snir, *Memory versus randomization in on-line algorithms.*, Proceedings 16th International Colloquium on Automata, Languages and Programming, (1989), pp. 687–703.

[14] D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules.*, Comm. ACM, 28 (1985), pp. 202–208.

[15] N. Young, *The k-server dual and loose competitiveness for paging*, Algorithmica, 11 (1994), pp. 525–41.

[16] N. Young, *Bounding the diffuse adversary*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 420–425.