On the Power of Randomization in Online Algorithms*

S. Ben-David
Technion, Haifa, Israel

A. Borodin†
University of Toronto, Toronto, Canada

R. Karp
University of California at Berkeley
and
International Computer Science Institute
Berkeley, California

G. Tardos‡
Eötvös University
Budapest, Hungary

A. Wigderson†
Hebrew University, Jerusalem, Israel

*A previous version of this paper appeared in the 22nd ACM STOC Conference Proceedings

†Part of this research was performed while these authors were visitors at the International Computer Science Institute

‡Part of this research was performed while this author was a visitor at the Hebrew University

Abstract

Against an adaptive adversary, we show that the power of randomization in online algorithms is severely limited! We prove the existence of an efficient "simulation" of randomized online algorithms by deterministic ones, which is best possible in general.

The proof of the upper bound is existential. We deal with the issue of computing the efficient deterministic algorithm, and show that this is possible in very general cases.

1 Introduction and Overview of Results

Beginning with the work of Sleator and Tarjan [17], there has recently been a development of what might be called a Theory of Online Algorithms. The particular algorithmic problems analyzed in the Sleator and Tarjan paper are "list searching" and "paging", both well studied problems. But the novelty of their paper lies in a new measure of performance, later to be called the "competitive ratio", for online algorithms. This new approach, called "competitive analysis" in Karlin, Manasse, Rudolph and Sleator [11], seems to have first been motivated by earlier attempts to understand the behaviour of so-called self organizing or self adjusting data structures. But as evident in the discussion provided by Karlin, et al [11] the issue transcends particular problems in data structures or paging.

Briefly stated, competitive analysis attempts to finesse the issue of what request sequences are likely in such environments (ie. average case analysis but one that has to account for distributions that reflect phenomena such as "locality of reference") by taking the following pessimistic approach in analyzing the performance of an online algorithm: an online algorithm is good only if its performance on *any* sequence of requests is within some (desired) factor of the performance of the optimal offline algorithm. In particular, a good algorithm would certainly perform well in the presence of an unknown distribution.

Following these studies of specific algorithmic problems, Borodin, Linial and Saks [3] gave an abstract formulation (called task systems) and a formal definition for the study of this new measure. Manasse, McGeoch and Sleator [12] introduced another abstract formulation, called K-server problems. In both the task system and K-server models, an online player is presented with a sequence of requests which must be satisfied by choosing amongst an allowable set of moves, each move having some nonnegative cost. (Raghavan and Snir [16] call these "games with moving costs".)

We present a more general framework for studying online algorithms — the *request-answer games*. In such a game an adversary again makes a sequence of requests, which are served (answered) one at a time by the online algorithm. The added generality is that now an arbitrary real valued function determines the cost of any sequence of requests and answers. This framework includes previous ones (e.g. K-server games [12] and task systems [3]) as special cases.

An online algorithm is called *c-competitive* if cost(algorithm) $\leq$ c$\cdot$cost(adversary) + $O(1)$, for every possible request sequence that is generated.

The Borodin, Linial and Saks [3] and Manasse, McGeoch and Sleator [12] papers primarily dealt with deterministic online algorithms, in which case the definition of being c-competitive is a rather routine matter. However, it was soon realized (see Borodin, Linial and Saks [2], Raghavan and Snir [15], Fiat et al [7]) that randomization could possibly offer the online player significantly more power; or perhaps one should say that the adversary now has relatively less power since the moves of the online player are no longer certain. In the case of randomized online algorithms, costs are taken to be expected values of the associated random variables and the definition of competitiveness becomes a more subtle issue, depending primarily on the nature of the adversary.

The related results of Borodin, Linial and Saks [2] and Fiat et al [7] assume an *oblivious adversary* (following the terminology to be adopted here and in the revised version of Raghavan and Snir [16]).

Oblivious adversary: One who must construct the request sequence in advance (based only on the description of the online algorithm but before any moves are made!), but pays for it optimally.

The Fiat et al [7] paper provides a dramatic example of the advantage provided by randomization against this adversary. Namely, they show that for the paging or cache problem with a cache of size K (ie. the K-server problem on the uniform metric space), there is a randomized algorithm (relative to any oblivious adversary) which achieves a competitive ratio of $O(\log K)$. (An optimal ratio of H_K is developed in [14], where H_K is the Kth harmonic number.) On the other hand, every deterministic algorithm can at best achieve a ratio of K. (This is the lower bound demonstrated for any K-server problem by Manasse, McGeoch and Sleator [12].)

The K-server conjecture of Manasse, McGeoch and Sleator [12] (which states that for every K-server problem there is a *deterministic* online algorithm with competitive ratio K) is still open. Fiat, Rabani, and Ravid [8] have made substantial progress on this conjecture by showing that for every K server problem the competitive ratio is bounded by a (exponential) function of K. The "random walk" approach, initiated by [15] and further developed in [5], [1], gave $O(K)$-competitive *probabilistic* algorithms for a variety of K-server problems, and possibly works for all of them. Recently, Grove [10] has been able to prove that the Harmonic algorithm given in [15] is $O(2^K)$ competitive for every K server problem. The interesting thing about these algorithms is that they achieve the same performance even against the following, much stronger, *adaptive adversary.*

Adaptive online adversary: One who makes the next request based on the algorithm's answers to previous ones, but serves it immediately.

It is obvious that for deterministic algorithms, this adversary is equivalent to the oblivious one, since the algorithm's answers are completely predictable. To understand just how much randomization helps against it, we introduce a yet stronger adversary (see also Raghavan and Snir [16]).

Adaptive offline adversary: One who makes the next request based on the algorithm's answers to previous ones, but serves them optimally at the end.

As might be conjectured , this adversary is so strong, that randomization adds no power against it!

Theorem 2.1 *If there is a randomized algorithm that is α-competitive against any adaptive offline adversary then there also exists an α-competitive deterministic algorithm.*

On the other hand, we can relate the performance of randomized algorithms against the three types of adversaries.

Theorem 2.2 *If G is a c-competitive randomized algorithm against any adaptive online adversary, and there is a randomized d-competitive algorithm* *against any oblivious adversary, then G is a randomized $(c \cdot d)$-competitive algorithm against any adaptive offline adversary.*

Our Theorems 2.1 and 2.2 together imply a deterministic algorithm whose performance is not much worse than the probabilistic ones. It shows that the results of [1], [5], [10] have deterministic counterparts with at most quadratically worse performance. In particular, using Grove [10] we obtain the best known deterministic competitive ratio for an arbitrary K-server system.

Unfortunately, the proof of Theorem 2.1 only guarantees the existence of a deterministic algorithm, and there is no general way to construct it even when the probabilistic one is given. We attack this problem from two directions.

In the first, we show how to explicitly construct a deterministic algorithm in Theorem 2.1, for a class of games that includes all finite K-server games and task systems. We lose a bit on performance: rather than a c-competitive algorithm whose existence is guaranteed, we construct a $((1+\epsilon)c)$-competitive algorithm, for every $\epsilon \geq 0$.

In the second, we finesse Theorem 2.1 altogether by explicitly constructing a deterministic algorithm in Theorem 2.2 with the same performance as the guaranteed probabilistic one. This can be achieved whenever the proof of c-competitiveness of the assumed algorithm against an adaptive online adversary is based on a computable potential function. Observing that all known proofs have this nature, this assumption at present does not lose much generality.

Though the game we define is infinite, every play terminates in a finite number of moves (since our adversaries are restricted to generate finite request sequences). This allows our definitions to be consistent with most of the initial papers concerning competitive analysis. Alternatively, Raghavan and Snir [16] formulate the concept of competitiveness in terms of infinite games. They develop analogues of our Theorem 2.1 (using classical results concerning determinacy in infinite games — see Gale and Stewart [9], Martin [13]) and Theorem 2.2. Raghavan and Snir [16] discuss the relation between these two approaches; in particular, they give a sufficient condition for when the alternative definitions of competitiveness are equivalent.

2 Definitions and Results

We study the performance of online algorithms in the general framework of request-answer games. In this game an online algorithm has to answer a sequence of requests trying to minimize its cost (as determined by the sequence of requests and answers). The algorithm is online, in the sense that it answers each request before seeing the following requests, and without knowing how long the sequence is. In some games, not all answer sequences are allowed.

A *request-answer game* consists of a request set R, a finite answer set A, and the cost functions $f_n : R^n \times A^n \to \mathbf{R} \cup \{\infty\}$ for $n = 0, 1, \ldots$. Let f denote the union, over all nonnegative integers n, of the functions f_n. Let us fix such a game.

A *deterministic online algorithm* G is a sequence of functions $g_i : R^i \to A$ for $i = 1, 2, \ldots$. For any sequence of requests $\underline{r} = (r_1, \ldots, r_n)$ we define $G(\underline{r}) = (a_1, \ldots, a_n) \in A^n$ with $a_i = g_i(r_1, \ldots, r_i)$ for $i = 1, \ldots, n$. The *cost* of G on $\underline{r}$ is $c_G(\underline{r}) = f_n(\underline{r}, G(\underline{r}))$. We will compare this to the optimal cost for the same sequence of requests: $c(\underline{r}) = \min\{f_n(\underline{r}, \underline{a}) | \underline{a} \in A^n\}$.

In this paper α and β will mean linear functions $\alpha, \beta : \mathbf{R} \to \mathbf{R}$. (Some of the theorems generalize to nonlinear functions, but linear are the important ones). We call the deterministic algorithm G *α-competitive* if for every request sequence $\underline{r}$ we have $c_G(\underline{r}) \leq \alpha(c(\underline{r}))$. In case $\alpha(x) = dx + e$ for some $d > 0$, G is sometimes said to have *competitive ratio d*.

A *randomized online algorithm* G is a probability distribution over deterministic online algorithms G_x (x may be thought of as the coin tosses of the algorithm G). For any request sequence $\underline{r}$ the answer sequence $G(\underline{r})$ and the cost $c_G(\underline{r})$ are random variables. We call a randomized online algorithm *α-competitive* if for any $\underline{r}$ we have $E_x(c_{G_x}(\underline{r})) \leq \alpha(c(\underline{r}))$.

Since we require good performance of a competitive algorithm for every request sequence $\underline{r}$ we can think of $\underline{r}$ as being given by an adversary. The adversary serves the requests in the optimal way, so his cost is $c(\underline{r})$. To distinguish this kind of adversary from the following more powerful adversary we call it an *oblivious adversary*.

An *adaptive adversary* is one that makes requests depending on the algorithm's answers to previous requests. This adversary comes in two flavours, according to the way it serves its own requests. The *adaptive offline adversary* answers the requests optimally when the whole request sequence is known. The *adaptive online adversary* however answers every request as soon as he makes it, before the algorithm does. For deterministic algorithms adaptive adversaries are not more powerful than the oblivious ones since the algorithm's moves can be foreseen. But for randomized algorithms it is worth introducing the α-competitiveness against adaptive adversaries. We call the α-competitive algorithms *α-competitive against any oblivious adversary* for contrast.

An *adaptive offline adversary* Q is a sequence of functions $q_n : A^n \to R \cup \{\mathsf{stop}\}$, where $n = 0, 1, \ldots d_Q$ and q_{d_Q} only takes the value "stop". For a deterministic algorithm G and an adaptive adversary Q we define the *actual request and answer sequences* $\underline{r}(G,Q) = (r_1, \ldots r_n)$ and $\underline{a}(G,Q) = (a_1, \ldots, a_n)$ together with $n = n(G,Q)$ recursively with $r_{i+1} = q_i(a_1, \ldots, a_i)$ for $i = 0, 1, \ldots n-1$, while $\underline{a}(G,Q) = G(\underline{r}(G,Q))$ and $q_n(\underline{a}(G,Q)) = \mathsf{stop}$. Note that these objects are uniquely defined in the order $r_1, a_1, r_2, a_2, \ldots, r_n, a_n, n$. The value $n = n(G,Q)$ is bounded by d_Q for any G. We define the cost of the algorithm G against the adversary Q to be $c_G(Q) = f_n(\underline{r}(G,Q), \underline{a}(G,Q))$. The cost of the adaptive offline adversary Q against the algorithm G is $c_Q(G) = c(\underline{r}(G,Q))$.

An *adaptive online adversary* $S = (Q,P)$ is an offline adaptive adversary Q, supplemented with a sequence P of functions $p_n : A^n \to A$ for $n = 0, 1, \ldots, d_Q$. Since $\underline{r}(G,S)$ is independent of P, we have $\underline{r}(G,S) = \underline{r}(G,Q)$, $\underline{a}(G,S) = \underline{a}(G,Q)$, and $c_G(S) = c_G(Q)$. We can also define the *answer sequence of the adversary* S to be $\underline{b}(G,S) = (b_1, \ldots, b_n)$ where $n = n(G,Q)$ and $b_{i+1} = p_i(a_1, \ldots, a_i)$ for $i = 0, \ldots, n-1$. We define the cost of S against the algorithm G to be $c_S(G) = f_n(\underline{r}(G,S), \underline{b}(G,S))$.

We define all these sequences and costs for a randomized algorithm G. In this case all these objects will be random variables.

We call a randomized algorithm G α-competitive against any offline (respectively online) adaptive adversary if for any offline adaptive adversary Q (respectively online adaptive adversary S) we have $E_x(c_{G_x}(Q)) \leq E_x(\alpha(c_Q(G_x)))$ (respectively $E_x(c_{G_x}(S)) \leq E_x(\alpha(c_S(G_x))))$. Note that α commutes with E_x.

Remark We imposed the requirement that the answer set is finite and the number of requests an adaptive adversary can put to one algorithm is bounded to ensure that all the expected values exist. Thus for any given randomized algorithm and adversary there are only finitely many possible request and answer sequences, and therefore the expected values are just weighted averages. It is possible to ensure the same by having an infinite answer set, but for any request declaring only finitely many answers "valid".

Our first theorem says that the adaptive offline adversary is so strong, that randomization doesn't help against it.

Theorem 2.1 *If there is a randomized strategy that is α-competitive against any offline adaptive adversary then there also exists an α-competitive deterministic algorithm.*

Proof: Consider the request-answer game as a two-person game between two players R and A such that in every step R gives A a request which A answers. A position in the game is a pair *($\underline{r}$,$\underline{a}$)*. Call a position *immediately winning for R* if $f_n(\underline{r}, \underline{a}) > \alpha(c(\underline{r}))$. Call a position $(\underline{r}, \underline{a})$ *winning for R* if there exists an adaptive rule for selecting requests, and a positive integer t such that, from the starting position $(\underline{r}, \underline{a})$, an immediately winning position for R will be reached within t steps regardless of how A plays. In particular, the initial position, in which $\underline{r}$ and $\underline{a}$ are both the empty string, is winning for R if and only if there exists an adaptive offline adversary Q such that, for any deterministic algorithm G, $c_G(Q) > \alpha(c_Q(G))$.

Suppose for the purpose of contradiction that R has a winning strategy corresponding to Q. If G is a randomized algorithm distributed over deterministic algorithms G_x then, taking the expected value of this inequality over all the choices of G_x, one obtains $E_x(c_{G_x}(Q)) > E_x(\alpha(c_Q(G_x)))$, which gives $E(c_G(Q)) > E(\alpha(c_Q(G))$. Therefore, no randomized algorithm can be α-competitive against the offline adaptive adversary Q. This contradicts our assumption that there exists a randomized algorithm that is α-competitive against any adaptive offline adversary. It follows that R does not have a winning strategy.

To complete the proof, we show that, if R does not have a winning strategy, then A must have a winning strategy; i.e., a deterministic algorithm that is α-competitive against every adaptive offline adversary. Note that a position *($\underline{r}$, $\underline{a}$)* is a winning position for R if and only if there exists a request r_{n+1} such that, for every answer a_{n+1}, $(\underline{r}r_{n+1}, \underline{a}a_{n+1})$ is again a winning position for R. (The validity of the 'if' part of this statement depends on the finiteness of A. For, if A were infinite, it might be the case that, although each of the infinitely many positions $(\underline{r}r_{n+1}, \underline{a}a_{n+1})$was winning for R, there would be no fixed upper bound on the number of steps needed to force an immediately winning position from the starting position $(\underline{r}, \underline{a})$, and hence no (finite) adversary would be able to force a win from that position.) Hence, if *($\underline{r}$, $\underline{a}$)* is not a winning position for R then for every request r_{n+1} there exists an answer a_{n+1} which is not a winning position for R. Thus, if any position is not winning for R, A can counter any request by R with an answer that will lead to another position that is not winning for R; it follows that, if A plays in this manner, an immediately winning position for R will never be reached. Thus, there is a winning strategy for A. ♣

Next we relate the power of the three kinds of adversaries.

Theorem 2.2 *Suppose G is α-competitive against any online adaptive adversary and there is a β-competitive randomized algorithm against any oblivious adversary. Then G is $\alpha \circ \beta$-competitive against any offline adaptive adversary.*

Proof: Fix any adaptive offline adversary Q, and assume G is distributed over deterministic algorithms G_x. Our task is to prove $E_x[c_{G_x}(Q)] \leq E_x[\alpha(\beta(c_Q(G_x))]$.

Let H be a randomized algorithm which is β-competitive against any oblivious adversary. If y denotes the coin flips of H, i.e. $H = \{H_y\}$, then we have for every $n, \underline{r} \in R^n$, $E_y(c_{H_y}(\underline{r})) \leq \beta(c(\underline{r}))$.

For each fixed y, define an adaptive online adversary $S_y = (Q, P_y)$ in such a way that for any deterministic online algorithm F, $\underline{b}(F, S_y) = H_y(\underline{r}(F, Q))$. This is a very simple-minded adaptive online adversary, who satisfies his own requests according to H_y and independently of the answers of the online algorithm F (i.e. all functions $(p_y)_i$ are constants). Intuitively, G is α-competitive against this online adversary which itself (when considered as an algorithm) is β -competitive against any offline adversary.

As G is α-competitive against adaptive online adversaries, we have that for every fixed y, $E_x[c_{G_x}(S_y)] \leq E_x[\alpha(c_{S_y}(G_x))]$, and taking expectations w.r.t y gives $E_y E_x[c_{G_x}(S_y)] \leq E_y E_x[\alpha(c_{S_y}(G_x))]$.

For every y note that $r(G_x, S_y) = r(G_x, Q) = r_x$. Then

$$
\begin{aligned}
& E_x[c_{G_x}(Q)] = E_y E_x[c_{G_x}(S_y)] \leq E_y[\alpha(E_x[c_{S_y}(r_x)])] \\
= \;& \alpha(E_x E_y[c_{H_y}(r_x)]) = \alpha(E_x[c_H(r_x)]) \leq \alpha(E_x[\beta(c(r_x))]) \\
= \;& E_x[\alpha(\beta(c_Q(G_x)))]
\end{aligned}
$$

♣

The algorithm RANDOM for the K server paging or cache problem shows that the bound of Theorem 2.2 is best possible in general. For as observed by Karlin (see Raghavan and Snir [16]), d is H_K for paging (McGeoch and Sleator [14]), c is K for RANDOM against any adaptive on line adversary (Raghavan and Snir [15]) and the optimal adaptive off line adversary can force a ratio of KH_K (Karlin).

In fact, Theorem 2.2 is tight in the following stronger sense: for any pair of positive numbers α and β with $1 \leq \beta \leq \alpha$, and any C less than $\alpha\beta$, one can construct a request-answer game such that

- there is an algorithm G that is α-competitive against any online adaptive adversary and β-competitive against any oblivious adversary;
- for every algorithm K, there is an adaptive off-line adversary against which K's competitive ratio is at least C.

Given α, β and C, the request-answer game is defined in terms of a positive integer parameter t, and positive real quantities m and M determined by the following pair of simultaneous equations:

$$\beta = \frac{(2t-2)m + M + 1}{2t}$$

$$\alpha = \frac{1 + (2t-1)M}{2 + (2t-2)m}$$

. The parameter t is chosen sufficiently large that $M \geq \max(m^2, C)$. This is possible since, by inspection of the equations, we see that, as t tends to infinity, m tends to β amd M tends to $\alpha\beta$. The request-answer game is specified as follows:

- The request set R is equal to the answer set A. Each of these sets consists of t disjoint pairs of elements. The two elements of any pair are called *mates*.
- The cost of the request-sequence-answer sequence pair $(r_1, r_2, \ldots, r_n)$, $(a_1, a_2, \ldots, a_n)$ is completely determined by a_1 and r_2, as follows: if $a_1 = r_2$ then the cost is 1; if a_1 is the mate of r_2 then the cost is M; otherwise, the cost is m.

The algorithm G simply draws its first answer, a_1, from the uniform distribution over A; its other answers are irrelevant. To see that G is β-competitive against any oblivious adversary note that, no matter how the adversary chooses r_2, G's cost will be 1 with probability $\frac{1}{2t}$, M with probability $\frac{1}{2t}$ and m with probability $\frac{2t-2}{t}$, giving an expected cost of $\frac{(2t-2)m+M+1}{2t}$, which is equal to β. Since the oblivious adversary's cost is at least 1, G is β-competitive. To see that G is α-competitive against any adaptive on-line adversary, note that, regardless of how the adversary chooses its first answer b_1, there will be exactly a $\frac{1}{2t}$ chance that $a_1 = b_1$ and a $\frac{1}{2t}$ chance that a_1 and b_1 will be mates. A simple case analysis shows that the adversary does best to choose r_2 equal to a_1 when $a_1 = b_1$, and to the mate of a_1 in all other cases. With this policy the adversary's expected cost per step is $\frac{2+(2t+2)m}{2t}$ and G's expected cost per step is $\frac{1+(2t-1)M}{2t}$, giving a competitive ratio of $\frac{1+(2t-1)M}{2+(2t-2)m}$, which is equal to α. Finally, regardless of how an on-line algorithm chooses a_1, an adaptive off-line algorithm can set its first answer, b_1, and its second request r_2, equal to the mate of a_1. Thus, the algorithm's cost will always be M and the adversary's cost will always be 1, giving a competitive ratio of M, which is at least C.

Corollary 2.1 *If there exists an α-competitive randomized strategy against any adaptive online adversary and a β-competitive randomized online strategy against any oblivious adversary, then there exists an $\alpha \circ \beta$-competitive deterministic strategy.*

Corollary 2.2 *If there exists an α-competitive randomized strategy against any adaptive online adversary, then it is $\alpha \circ \alpha$-competitive against any adaptive offline adversary and thus there is a deterministic $\alpha \circ \alpha$-competitive strategy.*

Corollary 2.3 *Consider the metric space defined by arbitrarily placing n nodes on a circle. For any $K < n$, there is a deterministic $4K^2$ competitive algorithm for the K server problem defined on this metric space.*

Proof: This corollary follows immediately from Corollary 2.2 and the randomized $2K$ competitive algorithm of Coppersmith, et al. [5].

The corollaries above prove the *existence* of a good deterministic online algorithm. We now turn to the question of *constructing* a deterministic algorithm from given randomized ones. In general, Deng and Mahajan [6] show that Theorem 2.1 and Corollary 2.1 cannot be made constructive. In particular, they show that there is a request-answer game for which there is a 1-competitive randomized computable online strategy, but there is no α-competitive computable deterministic online algorithm for any $\alpha > 0$. However, the following sections show that in many important cases, there is a constructive version of Corollary 2.1.

3 A Constructive Version of Corollary 2.1

Definition 3.1 *Let G be a randomized online algorithm. (It helps to think of G as playing against an adaptive online adversary. After n steps, $\underline{r} \in R^n$ denotes the requests so far, $\underline{a} \in A^n$ the algorithm's answers, and $\underline{b} \in A^n$ the adversary's answers.) Call a family $\Phi = \{\Phi_n : R^n \times A^n \times A^n \to \mathbf{R}\}_{n \geq 0}$ an* augmented potential function *for a function $\alpha : \mathbf{R} \to \mathbf{R}$ and the randomized online algorithm G, if the following holds:*

1) $\Phi_0 = 0$

2) For every n and configuration $(\underline{r}, \underline{a}, \underline{b}) \in R^n \times A^n \times A^n$, $\Phi_n(\underline{r}, \underline{a}, \underline{b}) \leq \alpha(f_n(\underline{r}, \underline{b})) - f_n(\underline{r}, \underline{a})$.

3) For every n, and every configuration $(\underline{r}, \underline{a}, \underline{b}) \in R^n \times A^n \times A^n$, every $r_{n+1} \in R, b_{n+1} \in A$, and a_{n+1} distributed on A according to $g_{n+1}(\underline{r}r_{n+1}, \underline{a})$ we have
$E[\Phi_{n+1}(\underline{r}r_{n+1}, \underline{a}a_{n+1}, \underline{b}b_{n+1})] \geq \Phi_n(\underline{r}, \underline{a}, \underline{b})$.

We can think of an augmented potential function as being composed of a "residue part", $\alpha(f_n(\underline{r}, \underline{b})) - f_n(\underline{r}, \underline{a})$, minus a pure potential function which reflects the difference between the configurations of the online player and that of the adversary. Potential functions play an essential role in the analysis of deterministic and randomized online algorithms. Theorem 1 of Manasse, McGeoch and Sleator [12] suggests the following observation:

Lemma 3.1 *Algorithm G is α-competitive against any adaptive online adversary if and only if there exists an augmented potential function for α and G.*

Proof: Let G be distributed over deterministic algorithms $\{G_x\}$. For any fixed adaptive online adversary S, let $n_x = n(G_x, S), r_x = \underline{r}(G_x, S), a_x = \underline{a}(G_x, S)$ and $b_x = \underline{b}(G_x, S)$.
if: Let $\Phi = \{\Phi_n\}$ be a potential function for α and G. First observe that for

every S, $E_x[\Phi_{n_x}(\underline{r_x}, \underline{a_x}, \underline{b_x})] \geq 0$. This follows from property (3) and induction on d_Q when $S = (Q, P)$. Now fix S. From property (2) we get

$$\begin{array}{rl} & E_x[c_{G_x}(Q)] - \alpha(E_x[c_Q(G_x)]) \\ = & E_x[f_{n_x}(\underline{r_x}, \underline{a_x}) - \alpha(f_{n_x}(\underline{r_x}, \underline{b_x}))] \\ \leq & -E_x[\Phi_{n_x}[\underline{r_x}, \underline{a_x}, \underline{b_x}]] \\ \leq & 0. \end{array}$$

only if: Assume G is α-competitive against online adversaries. Informally, $\Phi(\underline{r}, \underline{a}, \underline{b})$ will be $\sup_S[C_G(S) - \alpha(C_S(G))]$ with costs updated as if the game starts at this configuration, as S ranges over all adaptive online adversaries that reach this configuration against G. ♣

Theorem 3.1 *If $\Phi = \{\Phi_n\}$ is an augmented potential function for α and a randomized online algorithm G, and H is a randomized online algorithm that is β-competitive against any oblivious adversary, then the following deterministic algorithm $M = \{m_n\}$ is $\alpha \circ \beta$-competitive against an adaptive offline adversary. Asssume we defined $k_1, k_2, \ldots, m_n$, (and hence $M(\underline{r})$ for all $\underline{r} \in R^n$), and let $\underline{r}' = \underline{r}t \in R^{n+1}$. Then $m_{n+1}(\underline{r}')$ is chosen to satisfy*

$$\begin{array}{rl} & E_y[\Phi_{n+1}(\underline{r}', M(\underline{r})m_{n+1}(\underline{r}'), H_y(\underline{r}'))] \\ \geq & E_y[\Phi_n(\underline{r}, M(\underline{r}), H_y(\underline{r}))] \end{array}$$

Proof: Note that $m_{n+1}(\underline{r}')$ exists, since we can construct an online adversary $S_{r',y} = (Q(\underline{r}'), H_y)$ which asks the sequence $\underline{r}'$ and serves it using H_y. Φ is a potential function for α and G.

By induction on n, the configuration $(\underline{r}, M(\underline{r}), H_y(r))$ is reachable when G plays against H_y. By property (3)

$$\begin{array}{rl} & E_x[\Phi_{n+1}(\underline{r}', M(\underline{r})(g_{n+1})_x(\underline{r}'), H_y(\underline{r}'))] \\ \geq & \Phi_n(\underline{r}, M(\underline{r}), H_y(\underline{r})) \end{array}$$

Taking expectations w.r.t y on both sides, $m_{n+1}(\underline{r}')$ can be chosen to be $(g_{n+1})_x(\underline{r}')$ for the value of x which maximizes the potential Φ_{n+1}.

This proves inductively that for each n, $\underline{r} \in R^n$, $E_y[\Phi_n(\underline{r}, M(\underline{r}), H_y\underline{r}))] \geq 0$, and by property (2) that M is a deterministic online algorithm that is α-competitive against the (randomized) adaptive online adversary $S = (Q, H)$.

As in the proof of Theorem 2.2, the fact that H is β-competitive against any oblivious adversary implies that M is $\alpha \circ \beta$-competitive against any adaptive offline adversary. Of course, since M is deterministic, there is no difference between oblivious and adaptive adversaries. ♣

Corollary 3.1 *In the statement of Theorem 3.1, if Φ is computable and if G and H are computable algorithms (i.e. for every configuration and request the algorithms answer is a computable probabilistic function), then M is computable.*

Corollary 3.1 is somewhat imprecise in that we have not specified a precise notion of computability (say, for real valued functions). We claim the corollary holds for any reasonable notion. The complexity of M (i.e. its next answer function) is obviously determined by the complexity of computing the expected value of the potential function relative to some fixed randomized algorithm H (which may be G itself). We claim that all potential functions presently used in the analysis of randomized online algorithms are indeed efficiently computable (for example, see Raghavan and Snir [15], Coppersmith, et al [5], Berman, Karloff and Tardos [1], Grove [10]). More specifically, for all of the above K server algorithms, when applied to a v node graph, the expected value of the given potential function can be computed with cost bounded by a low degree polynomial in v and K. In particular, using Grove's [10] proof, we can construct an efficient deterministic $O(4^K)$ competitive algorithm for all K server systems.

4 A Constructive Version of Theorem 2.1

In this section we assume that the cost functions f_n satisfy two special properties: *monotonicity* and *locality*. Monotonicity means that extending a request-answer sequence cannot cause the cost to decrease; more formally, the requirement is that for all $n, \underline{r} \in R^n, t \in R, \underline{a} \in A^n, b \in A$ we have $f_{n+1}(\underline{r}t, \underline{a}b) \geq f_n(\underline{r}, \underline{a})$. Locality means that, for every positive real number h, only finitely many request sequences are of cost less than or equal to h; more formally, for all h, $\{\underline{r} : c(\underline{r}) \leq h\}$ is finite. All K-server games and task systems satisfy the monotonicity property. All K-server games on finite graphs, or on infinite graphs of bounded degree with finite edge costs, can be formulated so as to satisfy the locality property.

Let $\underline{r}, \underline{a}$ and $\underline{r'}, \underline{a'}$ be elements of the union, over all n, of $R^n \times A^n$. The *discrepancy* at $(\underline{r}, \underline{a}, \underline{r'}, \underline{a'})$ is defined as

$$\delta((\underline{r}, \underline{a}), (\underline{r'}, \underline{a'})) = f(\underline{rr'}, \underline{aa'}) - f(\underline{r}, \underline{a}) - f(\underline{r'}, \underline{a'})$$

The diameter of the game F is defined as

$$\begin{aligned} D(F) \quad = \quad & \sup\{|\delta((\underline{r}, \underline{a}), (\underline{r'}, \underline{a'}))| : (\underline{r}, \underline{a}) \in \bigcup_n (R^n \times A^n) \\ & \text{and } (\underline{r'}, \underline{a'}) \in \bigcup_n (R^n \times A^n)\} \end{aligned}$$

The diameter puts an upper bound on how much the sequence of past requests and answers can affect the incremental cost of a request-answer sequence. The diameter is finite, for example, in K-server games on finite graphs.

Theorem 4.1 *Let F be a game with a finite diameter* D(F)*, a finite set* R *of possible requests and a computable cost function satisfying the monotonicity and locality properties. Assume there exists a randomized online algorithm that is α-competitive against every offline adaptive adversary. Then for every $\epsilon > 0$, there is a computable deterministic online algorithm that is $((1+\epsilon)\alpha)$-competitive against every offline adaptive adversary.*

Proof: Since there is a randomized online algorithm that is α-competitive against any offline adaptive adversary, Theorem 2.1 establishes that there is a deterministic online algorithm that is α-competitive against any offline adaptive adversary. For any positive real number H, let R_H be the set of all request sequences $\underline{r}$ such that, for every prefix $\underline{r'}$ of $\underline{r}$ such that $\underline{r'} \neq \underline{r}$, $c(\underline{r'}) \leq H$. By the locality property R_H is a finite set, and, by monotonicity and the computability of the cost function, and the finiteness of the request set R, the request sequences in R_H can be effectively listed. Thus, for each H, there is a computable deterministic online algorithm A_H that is α-competitive against any adaptive offline adversary that is required to choose its request sequence from the finite set R_H. For any request sequence $\underline{r} \in R_H$, let $A_H(\underline{r})$ be the answer sequence produced by algorithm A_H in response to $\underline{r}$.

The required $(1+\epsilon)\alpha$-competitive algorithm involves a parameter H given by $H = \frac{(2+\epsilon)D(F)}{\epsilon}$. The algorithm decomposes any request sequence $\underline{r}$ as the concatenation of subsequences $\underline{r}(1), \underline{r}(2), \cdots, \underline{r}(t)$, where $\underline{r}(1)$ is the longest prefix of $\underline{r}$ in R_H, $\underline{r}(2)$ is the longest prefix in R_H of the suffix of $\underline{r}$ obtained by deleting the prefix $\underline{r}(1)$, and so forth. The answer sequence produced by the algorithm is then $A_H(\underline{r}(1)), A_H(\underline{r}(2)), \cdots, A_H(\underline{r}(t))$. The algorithm operates by repeatedly simulating the algorithm A_H; however, as soon as the request sequence is no longer in R_H the algorithm starts over, as if it had not received any previous requests.

Let $c(i) = c(\underline{r}(i))$. Let $\underline{r} = \underline{r}(1), \underline{r}(2), \cdots, \underline{r}(t)$. By the properties of the diameter D(F), we have:

$$c(\underline{r}) \geq c(\underline{r}(1)) + \sum_{i=2}^{i=t}(c(\underline{r}(i)) - D(F))$$

On the other hand, by the α-competitiveness of A_H and the definition of the diameter D(F), the cost incurred by the algorithm is at most $\alpha(c(1)) + \sum_{i=2}^{i=t}(\alpha(c(i)) + D(F))$. Also, by the definition of the decomposition of $\underline{r}$ into $\underline{r}(1), \underline{r}(2), \cdots, \underline{r}(t)$, $c(i) \geq H, i = 1, 2, \ldots, t-1$. Putting the inequalities together, we find that the algorithm is $(1+\epsilon)\alpha$-competitive. ♣

Returning to the example of a K server problem on a finite graph (say with minimum distance = 1), we observe that A_H can be constructed initially with cost $O(K^{KH})$. Then the complexity of the resulting algorithm is dominated by the $O(Kn^2)$, $n \leq KH$, dynamic programming cost (see Chrobak, et al [4]) for computing each $c(\underline{r}')$.

5 Acknowledgements

We thank Sandy Irani for helping with the construction showing that Theorem 2.2 is tight.

References

[1] P. Berman, H.J. Karloff, and G. Tardos. A competitive three-server algorithms. First Annual ACM-SIAM Symposium on Discrete Algorithms, pages 280–290, Jan 1990.

[2] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. To appear in *JACM*.

[3] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. Proceedings of the 19th Annual ACM Symposium on Theory of Computing, pages 373–382, New York City, NY, May 1987.

[4] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. First Annual ACM-SIAM Symposium on Discrete Algorithms, pages 291–300, Jan 1990. To appear in *SIAM Journal on Discrete Mathematics*.

[5] D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir. Random walks on weighted graphs, and applications to on-line algorithms. Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pages 369–378, Baltimore, Maryland, May 1990.

[6] X. Deng, and S. Mahajan. Randomization vs computability in online problems. Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, May 1991.

[7] A. Fiat, R.M. Karp, M. Luby, L.A. McGeoch, D.D. Sleator, and N.E. Young. Competitive paging algorithms. Technical Report CMU-CS-88-196, School of Computer Science, Carnegie Mellon University, 1988. To appear in *Journal of algorithms*.

[8] A. Fiat, Y. Rabani, and Y. Ravid. Competitive K Server Algorithms. Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, pages 454–463, St. Louis, Oct. 1990.

[9] D. Gale and F.M. Stewart. Infinite games with perfect information. In W.H. Kuhn and A.W. tucker, Editors, *Contributions to the Theory of Games Vol. II, Annals of Mathematics Studies, 28*, pages 245–266. Princeton University Press, Princeton, New Jersey, 1953.

[10] E. Grove. The harmonic k-server algorithm is competitive. Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 1991.

[11] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[12] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for on-line problems. *Journal of Algorithms*, 11:208–230, 1990.

[13] D.A. Martin. Borel determinacy. *Annals of Math.*, 102:363–371, 1975.

[14] L.A. McGeoch and D.D. Sleator. A strongly competitive randomized paging algorithm. Technical Report CMU-CS89-122, School of Computer Science, Carnegie Mellon University, 1989. To appear in *Algorithmica*.

[15] P. Raghavan and M. Snir. Memory vs. randomization in on-line algorithms. In *ICALP*, Italy, July 1989. *16th ICALP, LNCS* 372, Springer-Verlag, 19687–703.

[16] P. Raghavan and M. Snir. Memory vs. randomization in on-line algorithms. Revised version of *ICALP* paper.

[17] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *CACM*, 28(2):202–208, 1985.