This version `Wed Oct 15 18:15:25 PDT 2003` , and still under construction.

Using Potentials to Prove Competitiveness

A common method used to prove competitiveness of an online algorithm (either deterministic or randomized) is through the use of a *potential*.

Informal Overview. A potential is a real number which represents the amount of "unused credit" possessed by the algorithm. Each time the algorithm incurs a cost, it must expend credit, subtracting from the potential, and each time the adversary incurs a cost, the potential increases by C times that cost. In the deterministic case, the potential must be non-negative at the end of the request sequence, while in the randomized case, the expected value of the potential must be non-negative at the end of the request sequence. In most situations, the actual requirement is that the potential is non-negative at all times, since we never know when the request sequence will end.

Potentials for Directed Graphs

We begin by introducing a simple use of potentials for a different problem. Let G be a weighted directed graph, whose vertices are V and whose edges are E. That is, each edge $e \in E$ has a weight $w(e) \in \mathbf{R}$, the real numbers. We allow weights to be negative. Let $s \in V$ be be a given *start* vertex. The weight of a path is defined to be the sum of the weights of its constituent edges. We assume that all vertices are reachable from s.

Non-Positive Path Problem. Is it true that all paths in G starting at s have non-positive weight?

We say that a function $\Phi : V \to \mathbf{R}$ is a *potential* for (G, s) if the following three conditions hold:

1. $\Phi(v) \geq 0$ for all $v \in V$
2. $\Phi(s) = 0$
3. $\Phi(u) \geq w(e) + \Phi(v)$ for all $e = (u, v) \in E$

Theorem 1 *There is a potential for (G, s) if and only if all paths in G starting at s have non-positive weight.*

Exercise 1 Prove Theorem 1.

Proof: [Of Theorem 1] Suppose that Φ is a potential. We prove the following claim by induction on n:

Claim: For any path σ in G starting at s of length n, let v be the last vertex in σ. Then $w(\sigma) \leq -\Phi(v)$.

If $n = 0$, the claim is trivial. Suppose $n > 0$. Let τ be the prefix of σ consisting of all but the last edge. Let u be the last vertex of τ. Let $e = (u, v)$ be the last edge

of σ. By the inductive hypothesis, $w(\tau) \leq -\Phi(u)$, while by the definition of a potential, $\Phi(u) \geq w(e)+\Phi(v)$. Combining these inequalities with the equality $w(\sigma) = w(\tau)+w(e)$, we verify the claim. Since $\Phi(v) \geq 0$ for all v, we can conclude, from the claim, that $w(\sigma) \leq 0$ for any path σ starting at s.

To prove the converse, assume that the weight of every path starting at s is non-positive. G cannot contain any positive weight cycle, else a positive weight path could be constructed by using a path from s to a vertex on the cycle, then going around the cycle as many times as necessary to obtain a positive path weight.

For any vertex v, define $\Phi(v) = -\max\{w(\sigma) : \sigma \text{ is a simple path from } s \text{ to } v\}$. We need to prove that Φ satisfies the definition of a potential. Since the weight of every path from s to v is non-positive, $\Phi(v) \geq 0$. Since there is a simple path from s to s of weight 0 (the trivial path), $\Phi(s) \leq 0$, which implies that $\Phi(s) = 0$.

To prove the third condition for a potential, we first note that $\Phi(v) \geq w(\sigma)$ if σ is any path from s to v, since, if σ is not simple, it must contain cycles. Deleting all cycles of σ leaves a simple path whose weight is not less than the weight of σ, since no cycle has positive weight. Let $e = (u, v)$ be any edge of G. Let τ be a path from s to u such that $\Phi(u) = -w(\tau)$. Let σ be the concatenation of τ with e. By the definition of Φ, we have that $\Phi(v) \geq -w(\sigma) = -(w(\tau) + w(e))$. The result follows immediately. □

Game Graphs. We define a *game graph* to be a 6-tuple (A, B, E, w, s, T) such that

1. A and B are disjoint
2. $G = (A + B, E)$ is a directed graph.
3. $E \subseteq A \times B + B \times A$. (That is, every edge is from A to B or from B to A.) The edges from A to B we call *A-edges*. The edges from B to A we call *B-edges*.
4. w is a weight function on E.
5. $s \in V$, the *start vertex*, and all vertices are reachable from s.
6. $T \subseteq V$, the *final vertices*.[1]

The Game. There are two players, the *A-player* and the *B-player*. There is a *pebble* which is initially placed on s. At each time step, if the pebble is on a vertex in A, the A-player moves it along an A-edge to a vertex in B, while if the pebble is on a vertex in B, the B-player moves it along a B-edge to a vertex in A. The game ends when the pebble reaches a vertex in T. Thus, during the play of the game, the pebble follows a path from s to some $t \in T$, or else the game never ends. *A wins* if the total weight of that path is non-positive, or if the game never ends. *B wins* if the pebble eventually reaches T and the weight of its path is positive.

Exercise 2 Consider the well-known board game, chess.[2] Assume that Black is the chess champion of the world; thus, White considers even a draw to be a victory. Describe the

[1] Vertices of T could be in either A or B.

[2] If you don't know what chess is, substitute go, checkers, or tic-tac-toe.

game graph corresponding to this situation. (Do not attempt to draw the game graph, which is larger than the entire known universe.)

Game Potentials. A potential for a game graph (A, B, E, w, s, T) is a real valued function Φ defined on V which satisfies the following conditions: [You have to fill in these conditions.]

Exercise 3 Fill in the details of the definition of a *game potential* in such a way that Theorem 2, below, is true.

We say that "A can win" means if there is some online algorithm $\mathcal{A}$ such that, if A uses that algorithm to make its moves, A wins.

Theorem 2 *The A-player can win if and only if there is a game potential for the game graph.*

Proof: □

Exercise 4 Prove Theorem 2, using your definition of game potential.

Competitiveness

We will analyze different definitions of competitiveness for each of the five models. These are *competitiveness against the optimal offline algorithm*, and *competitiveness against an adversary*.

Given an online algorithm $\mathcal{A}$, and given any request sequence ϱ, we have defined $cost_{\mathcal{A}}(\varrho)$ for $\mathcal{A}$ belonging to any one of the five models: deterministic, distribution of deteriministic (which is called *barely random* if the distribution is finite), behavioral, distributional, and mixed. (Recall that the *mixed model*, which includes *knowledge state algorithms*, does not yet appear anywhere in the archival literature.) In the case of the deterministic and distributional models, $cost_{\mathcal{A}}(\varrho)$ is a real number, while in the other models, it is a random variable. For any model, the expected value $E(cost_{\mathcal{A}}(\varrho))$ is defined and is a real number.

The following definition of competitiveness is valid for all five models. Let $C \geq 1$. We say that $\mathcal{A}$ is *C-competitive against the optimal offline algorithm*, or just *C-competitive*, if there exists a constant K (which does not depend on the choice of ϱ) such that

$$E(cost_{\mathcal{A}}(\varrho)) \leq C \cdot cost_{opt}(\varrho) + K \tag{1}$$

for any request sequence ϱ. (Of course, in the deterministic model and in the distributional model, the expected value notation can be removed.)

Competitiveness Against an Adversary

The concept of an *adversary* was invented in order to better deal with claims of worst case performance of an algorithm. If $\mathcal{A}$ is any algorithm, and we are making the claim that, in the worst case, $\mathcal{A}$ achieves a certain goal, *e.g.*, satisfaction of inequality (1) above, it

sometimes helps to presume that the inputs are being given to $\mathcal{A}$ by an agent, which we call the *adversary*, which is consciously trying to defeat $\mathcal{A}$'s attempt to achieve the goal.

Just as we sometimes restrict the scope and power of $\mathcal{A}$, we sometimes restrict the scope and power of the adversary. The algorithm's power could be restricted by requiring it satisfy certain bounds on time or space complexity, for example, while the scope of an online algorithm is, by definition, restricted by the fact that it cannot see future inputs.

For an online problem, an adversary *adv* must both provide the request sequence ϱ as well its own solution, whose cost we denote as $cost_{adv}(\varrho)$. Since *adv* could itself be randomized, or could be interactive with $\mathcal{A}$, $cost_{adv}(\varrho)$ is a random variable.

Given $\mathcal{A}$ belonging to any of our five models, and given any specific adversary *adv*, we then say that $\mathcal{A}$ is C-competitive against *adv* if there exists a constant K such that

$$E(cost_{\mathcal{A}}(\varrho)) \leq C \cdot E(cost_{adv}(\varrho)) + K$$

where ϱ is the request sequence generated by *adv*. Since we permit *adv* to be randomized as well, ϱ is itself a random variable of type *string of requests*.

Adversary Models

The *strong* adversary knows essentially everything. Specifically, the strong adversary knows $\mathcal{A}$, and if $\mathcal{A}$ uses randomization, knows the outcome of every randomized choice that $\mathcal{A}$ might make.

The *oblivious* adversary knows everything about $\mathcal{A}$'s code, but cannot predict the outcome of any randomized choice that $\mathcal{A}$ might make. The oblivious adversary, in fact, is never told what $\mathcal{A}$'s configuration is: it's only job is to create the request sequence ϱ and also find its own solution.

The *adaptive online adversary* knows everything about $\mathcal{A}$'s code, and cannot predict the outcome of any randomized choice that $\mathcal{A}$ might make, although it knows those choices after they are made. This adversary must, however, service ϱ online. Specifically, the adversary announces the request r^t, then must choose its own configuration to service r^t. After making the choice of its own service, the adaptive online adversary is informed of $\mathcal{A}$'s choice.

Deterministic and Randomized Adversaries

Informally, a deterministic adversary chooses a request sequence and its own service sequence by using a deterministic algorithm, while a randomized adversary uses randomization.

Let $\varrho = r^1 \ldots r^n$ be the request sequence chosen by a given adversary *adv*, and let $\alpha = a^1 \ldots a^n$ be *adv*'s own service sequence.

Deterministic adversaries of each class are defined as follows.

- A deterministic oblivious adversary simply chooses one request sequence ϱ, and chooses α to be the optimal service sequence for ϱ.
- A deterministic strong adversary examines $\mathcal{A}$'s code, and also has access to $\mathcal{A}$'s random number generator. It deterministically chooses ϱ based on that knowledge, and chooses α to be the optimal service sequence for ϱ.

- A deterministic adaptive online adversary *adv* is in fact an online algorithm. Let h denote "halt," *i.e.*, a fictitious request that states that there will be no more requests. Given $r^i \in \mathcal{R}$ and $s^i, a^i \in \mathcal{S}$ for all $0 < i < t$, *adv* chooses r^t and a^t deterministically, where it is allowed that $r^t = h$. (We define $cost = 0$ if the "request" is h.)

We define a randomized adversary of a given class to be a distribution on the set of all deterministic adversaries of that class.[3]

The following theorem is a simplified version of Yao's Principle.

Theorem 3 *Given any $C \geq 1$ and any K, and given any online algorithm $\mathcal{A}$ of any of the five models, and given any of the three classes of adversaries, $E(cost_{\mathcal{A}}) \leq C \cdot E(cost_{adv}) + K$ for every randomized adversary adv in that class if and only if $E(cost_{\mathcal{A}}) \leq C \cdot E(cost_{adv}) + K$ for every deterministic adversary adv in that class.*

Proof: One direction of the proof is trivial, since every deterministic adversary is a randomized adversary with a trivial distribution. The other direction follows immediately from Theorem 2 of `ranvar.pdf`. □

"The" adversary. If we say that $\mathcal{A}$ is C-competitive against "the" adversary of a given class, we mean that there is some K such that $E(cost_{\mathcal{A}}) \leq C \cdot E(cost_{adv}) + K$ for every adversary of that class.

We summarize some of the known facts about competitiveness against the various adversaries. If $\mathcal{A}$ is an online algorithm, we define

- $C(\mathcal{A})$ = the competitiveness of $\mathcal{A}$ against the optimal offline algorithm.
- $C_{\text{strong}}(\mathcal{A})$ = the competitiveness of $\mathcal{A}$ against the strong adversary.
- $C_{\text{adon}}(\mathcal{A})$ = the competitiveness of $\mathcal{A}$ against the adaptive online adversary.
- $C_{\text{obl}}(\mathcal{A})$ = the competitiveness of $\mathcal{A}$ against the oblivious adversary.

Theorem 4

- $C(\mathcal{A}) = C_{\text{obl}}(\mathcal{A})$.
- $C_{\text{obl}}(\mathcal{A}) \leq C_{\text{adon}}(\mathcal{A}) \leq C_{\text{strong}}(\mathcal{A})$
- *If $\mathcal{A}$ is deterministic or distributional*[4]*, then* $C(\mathcal{A}) = C_{\text{strong}}(\mathcal{A}) = C_{\text{adon}}(\mathcal{A}) = C_{\text{obl}}(\mathcal{A})$.
- *If $\mathcal{A}$ is behavioral, then* $C_{\text{strong}}(\mathcal{A}) \leq C_{\text{obl}}(\mathcal{A}) \cdot C_{\text{adon}}(\mathcal{A})$

[3] Alternatively, a randomized adaptive online adversary can be defined using the behavioral model for online algorithms.

[4] My definition of the strong and online interactive adversaries in the distributional case differ from the typically understood definitions of the those adversaries. It is conventional to use the concept of the strong or online interactive adversary only for behavioral online algorithms. If $\mathcal{A}$ belongs to one of the other models, then $C_{\text{strong}}(\mathcal{A})$ conventionally means $C_{\text{strong}}(\mathcal{A}')$, and $C_{\text{adon}}(\mathcal{A})$ conventionally means $C_{\text{adon}}(\mathcal{A}')$, where $\mathcal{A}'$ is a behavioral online algorithm equivalent to $\mathcal{A}$ in the sense of Theorem 1 of the handout, "Models of Online Algorithms."

Potentials

We will give a definition of potential for each of the five models of online algorithms, except for distribution of deterministic. The following theorem obviously cannot be proved until we give those definitions:

Theorem 5 *If $\mathcal{A}$ is deterministic, behavioral, distributional, or mixed, then there is a C-potential of the same model as $\mathcal{A}$ if and only if $C \geq C_{\text{adon}}(\mathcal{A})$*

The Parameters of a Potential. The potential can have any parameters that make sense. It is normally a function of some combination of past actions, which could include the past request sequence and the current state of the online algorithm. In some cases, the potential is defined in terms of the future request sequence. Naturally, in this case, the online algorithm does not "know" the potential. Thus, the primary purpose of a potential is to help prove competitiveness, not to help the algorithm make decisions.

Deterministic Potentials

A *C-potential* (we just say "potential" if C is understood) for a deterministic online algorithm $\mathcal{A}$ is some real-valued function Φ, whose value at time t we express as Φ^t, such that, for every request sequence $\varrho = r^1 \ldots r^n$: $\Phi^t \geq 0$ for all $0 \leq t \leq n$ and $\Phi^{t-1} + C \cdot cost^t_{opt} \geq cost^t_{\mathcal{A}} + \Phi^t$ for all $0 < t \leq n$.

Theorem 6 *Let $\mathcal{A}$ be a deterministic online algorithm.*

1. *If there is a C-potential for $\mathcal{A}$, then $\mathcal{A}$ is C-competitive.*
2. *If $\mathcal{A}$ is C-competitive, then there is a C-potential for $\mathcal{A}$.*

The problem of the existence of a C-competitive deterministic online algorithm can be easily reduced to the problem of whether the A-player can win a game:

Exercise 5 Suppose we are given an online optimization program where $\mathcal{S}$, the set of configurations, and $\mathcal{R}$, the set of requests, are both finite. We are also given numbers $C \geq 1$, and K.

Construct a game graph $G = (A, B, E, w, s, T)$ such that the A-player can win G if and only if there exists a deterministic online algorithm $\mathcal{A}$ such that $cost_{\mathcal{A}}(\varrho) \leq C \cdot cost_{opt}(\varrho) + K$ for every request sequence ϱ.

Exercise 6 Prove part (1) of Theorem 6.

Hint: There is a close relationship between Theorem 6 and Theorem 2.

Behavioral Potentials.

Let $\mathcal{A}$ be a behavioral randomized online algorithm. Let $\mathcal{M}$ be the set of memory states of $\mathcal{A}$, and let Let $\mathcal{F} \subseteq \mathcal{S} \times \mathcal{M}$ be the set of all reachable full states of $\mathcal{A}$. We define a full state (s, m) of $\mathcal{A}$ to be *reachable* if there is some request sequence ϱ such that the probability that the full state of $\mathcal{A}$ will be (s, m) after serving the request sequence ϱ is positive. Let $C \geq 1$. We define a C-potential for $\mathcal{A}$ to be a function $\Phi(s, m, a) \geq 0$ defined for every $(s, m) \in \mathcal{F}$ and every configuration a, which satisfies the following conditions.

- $\Phi(s, m, a) \geq 0$ for all $(s, m) \in \mathcal{F}$ and all $a \in \mathcal{S}$.
- for any $(s, m) \in \mathcal{F}$ and any $a \in \mathcal{S}$, and for any $r \in \mathcal{R}$, write $\mathcal{A}(s, m, r) = (s', m')$. Note that s' and m' are random variables. Then, for any $a' \in \mathcal{S}$:

$$\Phi(s, m, a) + C\,cost_r(a, a') \geq$$
$$E(cost_r(s, s')) + E(\Phi(s', m', a'))$$

Theorem 7 *A behavioral online algorithm $\mathcal{A}$ has a C-potential if and only if $C \geq C_{\mathrm{adon}}(\mathcal{A})$.*

Proof:

Suppose that $\mathcal{A}$ has a behavioral C-potential. We can conclude that $\mathcal{A}$ is C-competitive against the adaptive online adversary.

[Verification needed here. This is the "easy" half of the proof.]

Conversely, suppose that $\mathcal{A}$ is C-competitive against the adaptive online adversary.

For each $(s, m) \in \mathcal{F}$, each $a \in \mathcal{S}$, we will define $\Phi(s, m, a)$, the potential if the full state is (s, m) and the adversary configuration is a, as follows. We will first define $\Phi^u(s, m, a)$ for each integer $u \geq 0$. Informally, $\Phi^u(s, m, a)$ is the minimum credit $\mathcal{A}$ will need to survive u requests if it is at full state (s, m) and the adversary is at a.

- For all $(s, m) \in \mathcal{F}$ and $a \in \mathcal{S}$, let $\Phi^0(s, m, a) = 0$.
- For all $(s, m) \in \mathcal{F}$, $a \in \mathcal{S}$, and $u \geq 1$, let

$$\Phi^u(s, m, a) = \max_{r \in \mathcal{R}, a' \in \mathcal{S}} E\Big(cost_r(s, s') - C \cdot cost_r(a, a') + \Phi^{u-1}(s', m', a')\Big)$$

  where $(s', m') \in \mathcal{F}$ is the random variable $\mathcal{A}(s, m, r)$.

- For all $(s, m) \in \mathcal{F}$ and $a \in \mathcal{S}$, let

$$\Phi(s, m, a) = \lim_{u \to \infty} \Phi^u(s, m, a)$$

- Let $K = \Phi(s^0, m^0, s^0)$.

We will now verify that Φ satisfies the definition of a behavioral potential.

[Verification needed.]

□

The Competitiveness of RANDOM for the k-Cache Problem

In the proof given below, will define a behavioral k-potential that appears to be quite "arbitrary." One of the things that baffles students is that proofs frequently contain mysterious choices that just "happen" to work out. During the lecture, I will explain that the choice of Φ is far from arbitrary; indeed, its value follows from a rather simple assumption (minimality) and is not difficult to calculate.

Theorem 8 RANDOM *is k-competitive for the k-cache problem, for any $k \geq 2$.*

Proof: RANDOM is described using the behavioral model, and we will define a potential using that model. Thus, we will actually prove that the competitiveness of RANDOM is at most k against the adaptive online adversary. A separate proof that we did in class showed that the competitiveness of RANDOM is at least k against the oblivious adversary, so the issue of the competitiveness of RANDOM against any of our adversaries is settled.

Our technique is to define a behavioral k-potential for RANDOM. Recall that RANDOM is memoryless, so we can neglect the memory parameter of the potential. For any $d > 0$, define $\Phi_d = kH_d$, where $H_d = 1 + \ldots + \frac{1}{d}$ is the harmonic number. Then, for any configurations S and T, define $\Phi(S,T) = \Phi_d$, where $d = ||S,T||$, the number of pages S that are not in T. (S denotes RANDOM's cache and T denotes the cache associated with an arbitrary service of ϱ, *i.e.*, the adversary's service.)

Let r be any request. Let $S' = \text{RANDOM}(S,r)$ be the random variable which chooses the next configuration of RANDOM, given that the next request is r and the current configuration is S. Recall that $S' = S$ if $r \in S$, while, if $r \notin S$, $S' = S - b + r$ for some $b \in S$ chosen at random using the uniform distribution. We will define T to be the adversary's cache before the request r, and we will assume that $T' = T - a + r$ is the adversary's cache after servicing that request, for some $a \in T$, where $a = r$ if $r \in T$.

Let $C = S \cap T$, let $B = S - C$, $A = T - C$, and D the set of all pages not in $S \cup T$. Let $d = ||S,T||$, which equals the cardinality of A and also of B. To shorten the notation, we write $\Phi = \Phi(S,T) = kH_d$, $d' = ||S',T'||$, and $\Phi' = \Phi(S',T') = kH_{d'}$.

All we need to do now is to verify that Φ satisfies the the definition of a behavioral k-potential. In the case of RANDOM, that inequality becomes:

$$\Phi + k||T,T'|| \quad \geq \quad E(||S,S'||) + E(\Phi') \tag{2}$$

By Theorem 7 we will then be done.

We break the verification of inequality (2) into cases. In each case, $\Phi = kH_d$.

Case 1: ($r \in C$). In this case, $a = b = r$, $cost_{\text{RANDOM}} = cost_{adv} = 0$, and $\Phi' = kH_d$. Thus, both sides of inequality (2) are the equal, and we are done.

Case 2: ($r \in A$). Then $cost_{adv} = 0$ and $cost_{\text{RANDOM}} = 1$. We have $b \in B$ with probability $\frac{d}{k}$, in which case $d' = d - 1$ and $\Phi' = kH_{d-1}$. With probability $\frac{k-d}{k}$, $b \in C$, in which case $d' = d$ and $\Phi' = kH_d$. It follows that $E(\Phi') = \frac{d}{k}kH_{d-1} + \frac{k-d}{k}kH_d = kH_d - 1$. Thus, both sides of inequality (2) are the equal, and we are done.

Case 3.1: ($r \in B$ and $a \in A$). Then $cost_{adv} = 1$, $cost_{\text{RANDOM}} = 0$, and $d' = d - 1$. The left side of inequality (2) is larger than the right side by $k + \frac{k}{d}$, and we are done.

Case 3.2: ($r \in B$ and $a \in C$). Then $cost_{adv} = 1$, $cost_{\text{RANDOM}} = 0$, and $d' = d$. The left side of inequality (2) is larger than the right side by k, and we are done.

Case 4.1: ($r \in D$ and $a \in A$). Then $cost_{adv} = 1$ and $cost_{\text{RANDOM}} = 1$. With probability $\frac{d}{k}$, we have $b \in B$, in which case $d' = d - 1$ and $\Phi' = kH_{d-1}$; with probability $\frac{k-d}{k}$, $b \in C$, in which case $d' = d$ and $\Phi' = kH_d$. Thus $E(\Phi') = kH_d - 1$. The left side of inequality (2) is larger than the right side by k, and we are done.

Case 4.2: ($r \in D$ and $a \in C$). Then $cost_{adv} = 1$ and $cost_{\text{RANDOM}} = 1$. With probability $\frac{d+1}{k}$, we have $b \in B + a$, in which case $d' = d$ and $\Phi' = kH_d$; with probability $\frac{k-d-1}{k}$, $b \in C - a$, in which case $d' = d + 1$ and $\Phi' = kH_{d+1}$. Thus $E(\Phi') = kH_d + \frac{k-d-1}{d+!}$. The difference between the left side of inequality (2) and the right side is thus $(k-1)\frac{d}{d+!}$, which is 0 if $d = 0$, and is positive if $d > 0$. Thus, we are done.

This completes all cases. □

We remark that the potential Φ given in the proof above is not unique, *i.e.*, we could have proved the same result using any of a variety of different potential. This particular choice of Φ happens to have the feature that it is the *minimal* potential, *i.e.*, that if Ψ is some other behavioral k-potential for RANDOM, then $\Phi \leq \Psi$ at all times.

Here is an important result that shows a limitation of usefulness behavioral potentials.

Theorem 9 *For any k, and for any $C < k$, there is no behavioral C-potential for any behavioral online algorithm for the k-cache problem.*

Proof: Let $\epsilon = k - C$. We shall prove the following lemma:

Lemma 1 *There exists a constant K such that, for every $t > 0$, there exist [Under construction]*

□

Exercise 7 Fill in the missing parts of the proof of Theorem 8. (There now are no missing parts, so this exercise is moot.)

Exercise 8 Examining your proof of Theorem 8, you should come to the realization that an adversary that behaves optimally against RANDOM (*i.e.*, forces RANDOM to pay as much as possible in relation to the optimal cost) will never make certain moves. Describe what this optimal adversary will never do, and why.

Distributional Potentials

Let $\mathcal{A}$ be a distributional online algorithm. For a given $C \geq 1$, we define a C-potential for $\mathcal{A}$ to be a function Φ which satisfies the following conditions: For some fixed K which does not depend on the request sequence, and for any request sequence, is some real-valued function Φ, whose value at time t we express as Φ^t, such that, for every request sequence $\varrho = r^1 \ldots r^n$: $\Phi^t \geq 0$ for all $0 \leq t \leq n$ and $\Phi^{t-1} + C \cdot cost^t_{opt} \geq cost^t_{\mathcal{A}} + \Phi^t$ for all $0 < t \leq n$. where it is understood that $cost^t_{\mathcal{A}}$ is defined in the usual manner for a distributional algorithm, using the minimum transportation cost.

Theorem 10 *Let $\mathcal{A}$ be a distributional online algorithm. For a given $C \geq 1$, There is a distributional C-potential for $\mathcal{A}$ if and only if $\mathcal{A}$ is C-competitive.*

Recall, by Theorem 4, that a distributional online algorithm is C-competitive if and only if there is an equivalent behavioral algorithm which is C-competitive against the oblivious adversary.

An H_k-Competitive Randomized Algorithm for the k-Cache Problem

For any given k, there is a randomized online algorithm, PARTITION, which is H_k-competitive, against the oblivious adversary. (There is no randomized online algorithm that is less than k-competitive against the strong adversary, or even the adaptive online adversary.) However, there is no such algorithm known which is simple to describe, nor do I believe one exists.[5]

The Case $k = 2$

We begin by defining PARTITION for $k = 2$. It uses an unlimited number of "bookmarks," *i.e.*, memory of pages which were once in the cache and have been ejected. (Note that, however, we sometimes deliberately "forget" that a page has been in the cache.)

Here is a description of PARTITION for $k = 2$, defined as a behavioral online algorithm.

Initialization. There are two pages in the cache, and there are no bookmarks. One of those pages is the MRU page, the other the LRU page. (It does not matter which.)

One Request. Let r be the requested page.

1. If r is the MRU page, do nothing.
2. If r is the LRU page, then r becomes the MRU page, the MRU page becomes the LRU page, and all bookmarks are erased.
3. If r is a bookmarked page, the LRU page is ejected, the MRU page becomes the LRU page, r becomes the MRU page, and all bookmarks are erased.
4. If r is a new page, let m be the number of bookmarked pages. With probability $\frac{1}{m+2}$, the LRU is ejected, while with probability $\frac{m+1}{m+2}$, the MRU is ejected. The remaining page in the cache becomes the LRU, and r becomes the MRU. The ejected page is bookmarked.

Note that the space complexity of PARTITION for $k = 2$ is $\Theta(n)$ in the worst case, where n is the length of the request sequence.

Theorem 11 *For $k = 2$,* PARTITION *is $\frac{3}{2}$-competitive.*

[5] PARTITION, as defined here for $k = 2$, and as defined in general for larger k, requires unlimited memory. There is a variation of PARTITION, called EQUITABLE, which also achieves the competitiveness H_k, and which requires finite memory for any fixed k.

By Theorem 9, there is no proof of Theorem 11 using a behavioral potential. Instead, we will redefine PARTITION as a distributional online algorithm and use a distributional potential.

Distributional Version of PARTITION for $k = 2$

Let $\varrho = r^1 \dots r^n$ be any request sequence. Without loss of generality, that there are no consecutive identical requests. (Such "repeat" requests can be deleted from the analysis; if the algorithm encounters one, it keeps the same distribution.) Assume that the initial cache contains two pages, which we "fictitiously" label to be r^0 and r^{-1}. We define a distribution σ^t of configurations for all $t \geq 0$.

- For each $t \geq 1$, σ^t is the uniform distribution on the configurations $\{\{r^t, r^{t-1}\}, \dots \{r^t, r^{t-w}\}\}$ for some $1 \leq w \leq t+1$. We call w the *window size*. (Effectively, PARTITION remembers w of the past requests.) We refer to $r^t \dots r^{t-w}$ as the *active* pages, the others as *inactive* pages. If $w = 1$, we say that σ^t is *collapsed*.
- $\sigma^0(\{r^0, r^{-1}\}) = 1$, *i.e.*, is collapsed.
- If r^{t+1} is not one of the past w requests, where w is the window size after t steps, then the window size after $t+1$ steps is $w+1$.
- If r^{t+1} is one of the past w requests, where w is the window size after t steps, then σ^{t+1} is collapsed, *i.e.*, σ^{t+1} is $\{r^t, r^{t+1}\}$ with probability 1.

Proof:(of Theorem 11) We will prove that PARTITION is $\frac{3}{2}$-competitive against the optimal offline algorithm. Let $\varrho = r^1 \dots r^n$ be any request sequence. Let a^t be the cache of the optimal offline algorithm, servicing ϱ, after t steps.

We begin by giving a distributional potential, as follows. For any $0 \leq t \leq 1$, let $w = w^t$ be the window size after t steps, and let σ^t be PARTITION's distribution after t steps, Define the potential after t steps, Φ^t, as follows: If $\sigma^t(a^t) > 0$, let $\Phi^t = \frac{w-1}{w}$. In this case, we say that the optimal offline algorithm is *overt* after t steps. If $\sigma^t(a^t) = 0$, let $\Phi^t = \frac{w-1}{w} + \frac{3}{2}$. In this case, we say that the optimal offline algorithm is *covert* after t steps. We need to show that the inequality given in the definition of a distributional potential holds. In our case, the inequality that must hold for each $t > 0$ is:

$$\Phi^{t-1} + \frac{3}{2} \cdot cost^t_{opt} \geq cost^t_{\text{PARTITION}} + \Phi^t \tag{3}$$

To simplify the notation, we write $\Phi = \Phi^{t-1}$, $\Phi' = \Phi^t$, $w = w^{t-1}$, $w' = w^t$, $r = r^t$, $a = a^{t-1}$, $a' = a^t$, $cost = cost^t_{\text{PARTITION}}$, and $cost_{opt} = cost^t_{opt}$. We break the proof into cases. Case 1: $w' = 1$, and the optimal algorithm is overt after $t-1$ steps and is overt after t steps. In this case, $cost = \Phi = \frac{w-1}{w}$, $cost_{opt} \geq 0$, and $\Phi' = 0$. Thus, both sides of (3) are equal.

[Under Construction]

□

Exercise 9 Finish writing the proof of Theorem 11.